UNIVERSITY OF OSLO
Department of Informatics

Current unresolved challenges and issues in next generation cloud deployments in a virtual environment

Muhammad Adnan Malik

Network and System Administration

Oslo University College

June 17, 2013

Current unresolved challenges and issues in next generation cloud deployments in a virtual environment

Muhammad Adnan Malik

Network and System Administration
Oslo University College

June 17, 2013

Abstract

The cloud computing technology has recently gain allot of attention both in public and private sectors. The technology is expected to grow up dramatically in the next 3-5 years. The deployment of next generation clouds has increased in public and private sectors. There are plenty of those who want to deploy a cloud locally, in their datacenters. Cost, less space in datacenters, increase in utility bills, management and efforts to prepare system for cloud are well-known hurdles stopping to deploy a cloud locally.

Deploying a cloud in a virtual environment can play an important role to overcome these hurdles. Ubuntu cloud infrastructure is an easy, simple and fast approach to deploy a next generation open source cloud in a real environment. This thesis finds the challenges and issues while considering such an approach in a virtual environment. It also finds the possible reason(s) of occurrence, possible solution(s) to the found challenges and issues and implements the found solutions.

To make an easy cloud deployment, this thesis helps to prepare a fully automated virtual environment for deploying cloud. And also makes automation possible in a certain context. Most of the solution has also been automated into the scripts.

Contents

1 Introduction **6**
- 1.1 Motivation . . . 6
- 1.2 Problem Statement . . . 8
- 1.3 Approach . . . 9
- 1.4 Thesis Outline . . . 9

2 Background and literature **11**
- 2.1 Virtualization . . . 11
  - 2.1.1 Virtualization tools and commands . . . 12
- 2.2 Cloud Computing . . . 13
  - 2.2.1 Public and Private Clouds . . . 13
  - 2.2.2 Infrastructure as a service (IAAS) . . . 14
- 2.3 OpenStack . . . 14
  - 2.3.1 Components of OpenStack . . . 15
- 2.4 Ubuntu Cloud Infrastructure . . . 17
- 2.5 MAAS . . . 18
- 2.6 Juju and Juju Charms . . . 19
  - 2.6.1 Systems prepared in Ubuntu Cloud Infrastructure . . . 20

3 Approach **21**
- 3.1 Objectives . . . 21
- 3.2 Environment . . . 22
  - 3.2.1 Physical Server . . . 22
- 3.3 Approach for Deployment . . . 22
  - 3.3.1 MAAS and Juju . . . 23
  - 3.3.2 Infrastructure design . . . 23
- 3.4 Installation and preparation of the environment . . . 25
  - 3.4.1 Installing KVM and necessary softwares . . . 25
  - 3.4.2 Configuring bridge . . . 25
  - 3.4.3 Creating virtual Network . . . 26
  - 3.4.4 Creating virtual machine for MAAS . . . 26
  - 3.4.5 Creating virtual machine for deploying services . . . 27

4 Results **29**
- 4.1 challenges and issues . . . 29
  - 4.1.1 Challenge Wake on lan . . . 30
  - 4.1.2 Challenge Time out . . . 31

4.1.3 Issue 409 CONFLICT 33
4.1.4 Issue: Juju status 35
4.1.5 Issue: Cobbler Errors 36
4.1.6 Issue Pending ID 40
4.1.7 Issue fault code 99 40
4.1.8 Issue: block mapping 42
4.1.9 Challenge booting nodes 43
4.1.10 Issue: Authentication 45
4.1.11 Issue: Internal Server Error 46
4.1.12 Issue: MAAS pxe boot issue 47
4.1.13 Issue: relation error 48
4.1.14 Issue: Juju charm 49
4.1.15 Challenge: DHCP not found 49
4.1.16 Issue: Fixing Juju charm 49
4.2 Challenges summary 50
4.3 Issues summary 50
4.4 Automation 50
4.4.1 Automated virtual environment for Ubuntu cloud Infrastructure 51
4.4.2 Automated virtual machine for MAAS 51
4.4.3 Deploying MAAS and Juju and fixing some Issues 52
4.4.4 Creating VM for services deployment and fixing challenges and issues 53
4.4.5 Automats juju services deployment 53
4.4.6 Automats Starting nodes and fixing some challenges and issues 53
4.4.7 Creating Domain Controller and DHCP and fixes some challenges and issues 53

5 Analysis and Discussion **54**
5.1 Automation 54
5.1.1 Preparing environment 54
5.1.2 Virtual machine deployment 55
5.1.3 Installing and setting up MAAS and Juju 55
5.2 Challenges, issues 56
5.2.1 Challenge: Wake on lan 56
5.2.2 Challenge: Timed Out 57
5.2.3 ignored IPMI error 57
5.2.4 Issue: Issue 409 CONFLICT 58
5.2.5 Issue: Juju status 58
5.2.6 Issue: Cobbler Errors 58
5.2.7 Issue Pending ID 58
5.2.8 Issue fault code 99 59
5.2.9 Issue: block mapping 59
5.2.10 Challenge : booting nodes 60
5.2.11 Issue: Authentication 60
5.2.12 Issue: MAAS pxe boot issue 60
5.2.13 Issue: Internal Server Error 61
5.2.14 Issue: relation error 61
5.2.15 Issue: Juju charm 62
5.2.16 Issue: Fixing Juju charm 64

5.3 Automation after applying solutions 64

6 Conclusions **66**
6.1 Recommendations 67

7 Future work **68**

A builder.pl **69**

B network.sh **70**

C creater.sh **71**

D start.pl **72**

E approach1: local DNS **73**
E.0.1 Using Libvirts DHCP and DNS 73

F approach2: Separate domain controller **76**
F.0.2 Case 2: DHCP and DNS on a separate virtual machine . 76
F.0.3 Fixing DHCP challenge and continuing 77

List of Figures

2.1 Role of IAAS 15
2.2 openstack software diagram 16
2.3 Components of OpenStack software diagram 16
2.4 Comparing Ubuntu Cloud infrastructure 18

3.1 Infrastructure design 24

4.1 Unknown cobbler error. 37
4.2 cobbler timed out error 38
4.3 Mass dashboard error 41

5.1 MAAS IPMI error 57
5.2 DHCP and DNS at KVM libvirt 62
5.3 DHCP and DNS at seprate server 63

E.1 DHCP and DNS at KVM libvirt 74

List of Tables

3.1 Physical system specifications 22
3.2 Server end systems 24
3.3 specifications of system ″controller″ 27
3.4 Nodes specifications 28

E.1 Systems in case:1 74

F.1 Systems in case:2 77

Chapter 1

Introduction

1.1 Motivation

Today, changing the existing data centers infrastructure to cloud computing technology has become an important question for any organization. Utilizing the existing resources in the best possible way is a key demand in datacenters. Expense in terms of operations of servers, maintenance, cooling power and spaces are generally high, for any private organization [25][17][41]. Resource optimization is the key to minimize the cost. Adding additional resources is a demand for any growing business. Rather adding many hardware resources, adding a few powerful resources can help to avoid an increase in mentioned expenses.

Servers virtualization played an important role to in resource optimization and it is growing fast in the industries today [11][46]. Virtualization helps to manage the resources in a flexible way, such as that any change in existing infrastructure may not become a big hurdle. The more the business grows the greater resources are consumed, which lead to increase in number of physical servers in a datacenter. And so is the increase in the cooling costs of resources and energy bills. Virtualization can help by modifying the existing system in such a way that maximum efficiency is achieved from a physical system rather than using two separate physical systems for less CPU power consuming services. Virtualization can also assist to avoid growth of number of physical resources by adding less number of powerful physical resources. So rather adding more physical systems, one powerful system could be added. Such powerful system is then configured by using virtualization techniques to achieve maximum possible services in their own environment. Easy management of the system and fewer chances of hardware failures can be achieved by using virtualization. Disaster recovery and live migration is much easier and reliable in virtual environment than by using physical systems.

Cloud computing has recently seen a growth in a virtualized environment. Cloud using a virtual environment is more flexible and scalable solution for

any datacenter. An organization demands more resources in their peak seasons. Buying these additional resources for using only in peak seasons was the ultimate solution before the concept of cloud computing. These additional resources were not used during off season. Amazon came up with a smart idea of renting out the power or these additional resources during their off season. Amazon earned money and also helped other organizations in their peak seasons. Amazon named this idea as cloud computing. Today most of the businesses are getting advantages by using public clouds. There are various advantages of public clouds but privacy and less control over the systems are considered as major concern in a public cloud environment. Private cloud computing is recently been targeted to avoid increase in expenses and privacy problems. Private cloud gives more control on the resources but on demand, resources are considered to be a problem in terms of losing ownership of resources. On demand, resources are normally used during peak sessions and are paid only when they are used.

To decrease cost in a private cloud, maximum performance is tuned with minimum physical resource utilization. Virtualization techniques help to achieve this goal in an efficient way. By using nested virtualization at a high power system, data centers infrastructure can be easily converted into a private cloud hosting datacenter. The existing physical resources in the datacenter can be added to the cloud to increase its power. Virtual cloud deployment takes less efforts and risk to shift an existing datacenter into a private cloud. Building private cloud in an existing datacenter is a challenge for small companies and institutions. Cost, space, management and efforts in preparing a system for cloud, inside an existing datacenter are the major hurdles to move to private cloud network.

MaaS and Juju are promising emerging technologies, used by Ubuntu and Canonical, the leading open source operating system provider, to deploy OpenStack cloud on physical infrastructure. Ubuntu Cloud Infrastructure is one of the easiest way to deploy a cloud based on OpenStack. “Ubuntu Cloud Infrastructure is ready to deploy Infrastructure-as-a-Service (IaaS) based on OpenStack. It provides you all the tools you need to offer a private IAAS cloud on your own hardware”[65].It uses Metal as a service (MAAS) and Juju to setup an OpenStack based cloud. MAAS deals with raw metals or machines without installed operating system. MAAS made it very easy to deploy nodes and these nodes can be updated or retired from the system with a single click. MAAS turns on the machines itself by using wake-on-lan feature and prepares the machines for further use. Juju configures the machine and it deploys services on these machines using juju charm. Juju is a slang word that means magic[16].Juju uses charms to deploy the services on the machines prepared by MAAS. Charm store offers 209 different services and each of them can be deployed by a single command. Once the services are started, relations are created between these services and cloud is ready to be used. Ubuntu Cloud Infrastructure has made it very easier to scale the services. Services can be scaled up and down on the go. More compute nodes can be added very easily

to increase the power of the cloud.

Some effort has been done to find the challenges and issues while deploying such a system into a virtual environment. This thesis explains the current unresolved challenges issues in next generation cloud deployments in a virtual environment. Though it is said that ubuntu cloud infrastructure is called the simplest[65],easiest[42] and fastest[64][35] way to deploy a cloud or it has some challenges and issues, when it is deployed in a virtual environment. Technology does not work out of the box, when it is used in a different environment then directed. Step by step investigation has been made while deploying Ubuntu Cloud Infrastructure in a virtual environment. The thesis also focuses on finding and implementing the possible solutions to the discovered challenges and issues, with or without automation. Some scripts has been produced to implement automation at various steps.

1.2 Problem Statement

- What are the current challenges facing system administrators, who wants to deploy OpenStack with Ubuntu Cloud Infrastructure in a virtual environment? What could be the possible cause and solution of these challenges? How the possible solutions can be implement.
- What are the current issues in MAAS and Juju, while using them in a virtual environment? What could be the possible reasons and solution to these issues? How these issues can be fixed by implementing the possible solutions?

Canonical has produced Ubuntu Cloud Infrastructure for non-virtual environment. Technology does not work out of the box, when it is used in a different environment then directed. The first question is to find the challenges while deploying an OpenStack based cloud using Ubuntu Cloud Infrastructure in a virtual environment. What could be the possible reason(s) behind these challenges and what could be the possible solution to fix these challenges. And finally how the possible solution(s) can be implemented to fix these challenges in a virtual environment?

Ubuntu claims that Ubuntu cloud infrastructure is the simplest and easiest route to deploy an OpenStack Cloud[65][42]. It is also much faster than deploying a cloud manually[64][35]. So how easy, fast and simple it is to deal with MAAS and Juju in a virtual environment? The second question covers the issues in MAAS and Juju, finding possible solutions to these issues and implementing the possible solutions, in a virtual environment.

1.3 Approach

Challenges and Issues shall be discovered by following the official documentation to setup the cloud. If needed, help will be taken from other documentations to solve the challenges or issues.

MAAS and Juju shall be deployed on the first virtual machine and later more virtual machines shall be created for deploying various services. The machines created after setting up MAAS, shall be prepared by MAAS server for operating system installation and configuration. Juju will be later use juju charm to deploy OpenStack services on these virtual machines. After deploying the services, relations will be created between different services. Some services shall be exposed to login to the OpenStack dashboard.

A general approach will be to find challenges and issues an issue while doing above explained process. Possible solutions to fix these issues or challenges shall be found first and then these possible solutions shall be implemented. Only after fixing each challenge or issue, the next step shall be taken in the process of cloud deployment.

Maximum possible automation shall be implemented by creating some scripts.

1.4 Thesis Outline

This thesis is structured as following

- Chapter 1: This chapter introduces about the thesis by starting from motivation and continues towards describing the problem statement

- Chapter 2: Background information of each important topic will be covered here. It covers research information to achieve the goals.

- Chapter 3: Describes the goals and the approach that will be taken to achieve the goals. The Approach will explain the proposed solution to create the system design. It also explains the installation and configuring the environment.

- Chapter 4: Steps taken to find the challenges and issues shall be explained in this chapter. Results achieved by doing these experiments will be explained during the experiments and a summary of these results will be explained at the end of the chapter. It also covers the findings during the thesis and facts about the created scripts.

- Chapter 5: Analysis and discussion on achieved results will be carried out in this chapter. Future work and suggestions will be listed at the end of this chapter.

- Chapter 6: Explains the conclusion and summarization of the thesis.

Chapter 2

Background and literature

This chapter explains the services and tools used in this project. The sections are arranged in the order starting from explaining virtualization, cloud computing and technology used for cloud deployment, finally Infrastructure as service is explained and followed by Ubuntu cloud infrastructure. This chapter covers detailed explanation of important concepts of the used technology.

2.1 Virtualization

Virtualization can be defined as doing segmentation of a physical computer to achieve multiple concurrent operating systems execution. [72]. Servers virtualization played an important role to in resource optimization and it is growing fast in the industries day by day[11][46].

Virtualization helps to manage the resources in a flexible way, such as any change in existing infrastructure may not become a big hurdle. The more the business grows the better the resources are consumed, which lead to increase in the number of physical servers in a datacenter. [48]. And so is the increase in the cooling costs of resources and energy bills.

According to a recent estimate, sixty percent of small and medium size companies are using virtualization [51]. Virtualization can be helpful by modifying the existing system in such a way that maximum efficiency is achieved from a physical system rather than using two separate physical systems against less CPU power consuming services. Virtualization helps to maximize utilization of computing power and minimizes the configuration work and management issues [44]. Virtualizing clusters make it quite easier to deploy and manage big pools of virtual systems.

Virtualization can also help to avoid growth of number of physical resources by adding less number of powerful physical resources. So rather adding more physical systems, one powerful system could be added. Such powerful system is then configured by using virtualization techniques to achieve maximum possible services in their own environment. Easy management of the

system and fewer chances of hardware failures can be achieved by using virtualization. Disaster recovery and live migration is much easier and reliable in virtual environment than using physical systems. [**?**].

Virtualization assists in educational environment especially in IT education by using curriculum with virtualization methodologies. Studies from past few years have shown that using virtualization in educational institutions helps to increase the quality of education as well as decrease in hardware and software costs.[28][33][27][70][74][3][43][29][29]

Different pools of Virtual machines are created inside a cloud with the help of Hypervisors. Similarly building a cloud itself in a virtual environment is an example of nested virtualization.

2.1.1 Virtualization tools and commands

For managing a hypervisor, some commands and tools are utilized. Some of them are explained below. In a heterogeneous data center, it is often impossible to control with a single tool multiple virtualization solutions [47]. This project uses more than one tool to deal with virtualization operations. It is possible to work only with libvirt and virsh but for easy and quick network installation of operating system, VM-Builder has been chosen.

2.1.1.1 Libvirt and Virsh

Libvirt is an open source toolkit, used for interacting with/to virtualization operations. Libvirt uses XML files to manage virtual machines, virtual networks and storage. Each xml file contains full information about the virtual machine or network to be created. Libvirt is compatible with KVM/QEMU linux hypervisor. [32] [30]

Creating, deleting, running, stopping and managing virtual machines have become a lot easier by using simple virsh commands. It has become convenient for system administrator to implement automation scripts by using virsh [23]. Virtual machines using virsh are created by writing a libvirt XML file. [39]. It has set of short and easy commands to do operations. For example virsh start vm1. Command starts a virtual machine. Similarly commands like virsh destroy and virsh undefined deletes and removes the machines respectively [31][68].

2.1.1.2 VM-Builder

Vmbuilder is an automation script, used for creating virtual machines [59]. Vmbuilder is currently being developed by Ubuntu virtualization team. It supports the preparation of virtual machines by using Ubuntu online mirror. It is much easier to use vmbuilder to create a virtual machine by specifying all the required specification of the virtual machine, in a single command[50].

Virtual disk images can be defined in a single text file vmbuilder.partition before creating a VM using vmbuilder. Post installation of packages and setting password can be achieved easily by using vmbuilder. [52]

2.2 Cloud Computing

A recent survey has been carried out in America by Wakefield Research for Citrix which has discovered that 51 percent out of 1,000 people think that stormy weather can affect cloud computing. 29 percent of the participants said that cloud is an actual cloud. Only 16 percent knew the true meanings of cloud computing [69]. It is unclear that how many people know about cloud computing in formal sense, some believes, it is about large databases and some said email servers. Both are true up to some extent and some assumed that cloud technology is coming from actual clouds. [21]

One can understand this concept as: Building an infrastructure to combine computing power, and deploying applications and services on the infrastructures so that these services are available to you wherever and whenever you need it[22]. Buyya et al describes cloud computing as : a cloud is a type of parallel and distributed system ,consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements ,established through negotiation between the service provider and the consumers [73].

Exploring the history of cloud computing makes it quite easier to understand cloud computing. An organization demands more resources in their peak seasons. Buying these additional resources for using only in peak seasons was the ultimate solution before the concept of cloud computing. These additional resources were not used during off season. Amazon came up with a smart idea of renting out the power or these additional resources during their off season. Amazon earned money and also helped other organizations in their peak seasons. Amazon named this idea as cloud computing. [36] [9]. Today china has probably done the largest investment in cloud computing[14]. Government press releases across the world, states that investment in cloud computing is expected as dramatic increase in next three to five years [4] [5] [38] [26] [20].

2.2.1 Public and Private Clouds

Public cloud computing has become famous from past few years. Organizations can scale up and down the resources on the go by using cloud computing[71]. A lot of organizations are using public clouds to avoid many challenges like management of the hardware infrastructure and space on datacenters. There are various advantages of public clouds but privacy and less control over the systems are considered as major concern in a public cloud environment. Private cloud computing is recently been targeted to avoid increase in expenses

and privacy problems. Private cloud gives more control on the resources but on demand, resources are considered to be a problem in terms of losing ownership of resources. On demand, resources are normally used during peak sessions and are paid only when they are used.

To decrease cost in a private cloud, maximum performance is tuned with minimum physical resource utilization. Virtualization techniques help to achieve this goal in an efficient way. By using nested virtualization at a high power system, data centers infrastructure can be easily converted into a private cloud hosting datacenter. The existing physical resources in the datacenter can be added to the cloud to increase its power. Virtual cloud deployment takes less efforts and risk to shift an existing datacenter into a private cloud. Building private cloud in an existing datacenter is a challenge for small companies and institutions. Cost, space, management and efforts in preparing a system for cloud, inside an existing datacenter are the major hurdles to move to private cloud network.

2.2.2 Infrastructure as a service (IAAS)

Cloud computing includes different services e.g software as a service (SaaS), infrastructure as a service (IAAS), and platform as a service (PaaS). IAAS offers the best use of high speed metal devices to convert into a cloud by using hypervisors like KVM, Xen etc. An environment is built for each client such as one can install operating systems from the images and applications according to the needs. The pools are customizable according to the customer needs and provided on demand. Some very famous vendors providing IAAS are Amazon EC2, Google Compute Engine, HP Cloud and Rackspace Cloud.

Following diagram gives an easy understanding of various services controlled at IAAS and it also explains the location of IAAS in a cloud services hierarchy.

[2] This thesis focuses on infrastructure as a service (IAAS). The core component of this architecturethe cloud OSis responsible for managing and monitoring the physical and virtual infrastructures, providing abstraction of the underlying infrastructure, and offering different tools and advanced functionality for cloud users. [37]

2.3 OpenStack

There are various solutions to build a cloud e.g for example .OpenStack, CloudStack, Eucalyptus etc. Canonical uses OpenStack to build Ubuntu cloud Infrastructure. OpenStack is introduced by NASA and Rackspace. It is very similar to Amazon S3. But it is a community of different vendors. Rackspace Cloud Computing calls it is an massively scalable open source cloud Operating system[45]. OpenStack works at IAAS layer of computing, dealing with virtual machines and networks etc. Below diagram explains the concepts of OpenStack.

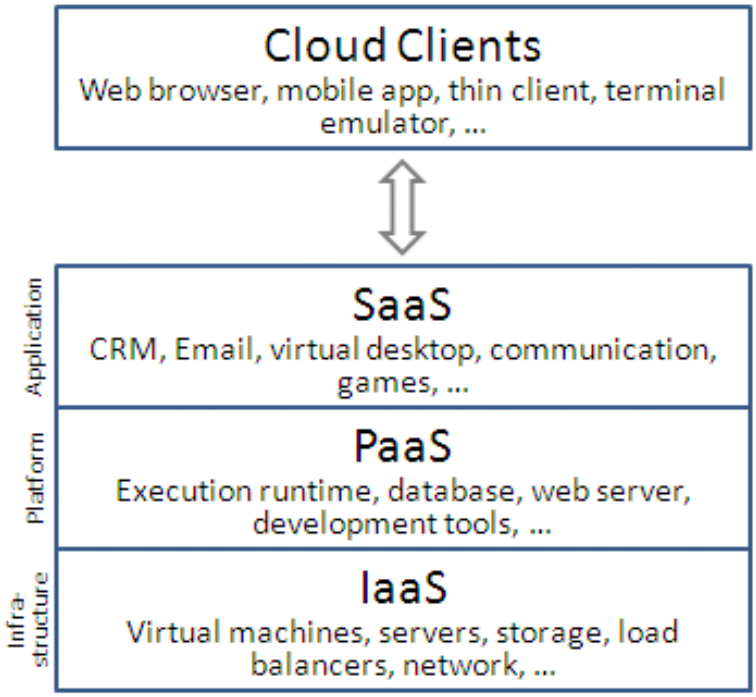

Figure 2.1: Role of IAAS
[2]

OpenStack components are deployed on a slandered hardware. These components are configured to interact with one and other for different operations, for example: for authentication, keystone service is contacted. The end user is not supposed to be worried about how these services are configured as long as the services are working fine. OpenStack dashboard is exposed to end user to drive the cloud according to their needs.

2.3.1 Components of OpenStack

OpenStack has different components to deal with different services. Figure 2.3[8] explains the way these components interact with one and other.

These components are explained briefly. [15]

- Compute (codenamed ”Nova”)

  It provides virtual servers on demand. It is known as cloud computing fabric controller and is the main part of IAAS.

- Object Store (codenamed ”Swift”)

  It is used to store and retrieve data but not mountable directories like a file server. Different companies are providing storage services using this concept. It is also used internally to store data.

- Identity (codenamed ”Keystone”)

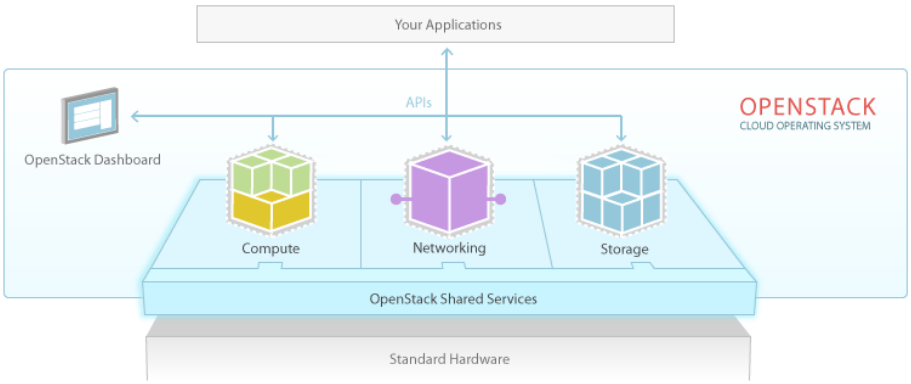

Figure 2.2: openstack software diagram
[40]

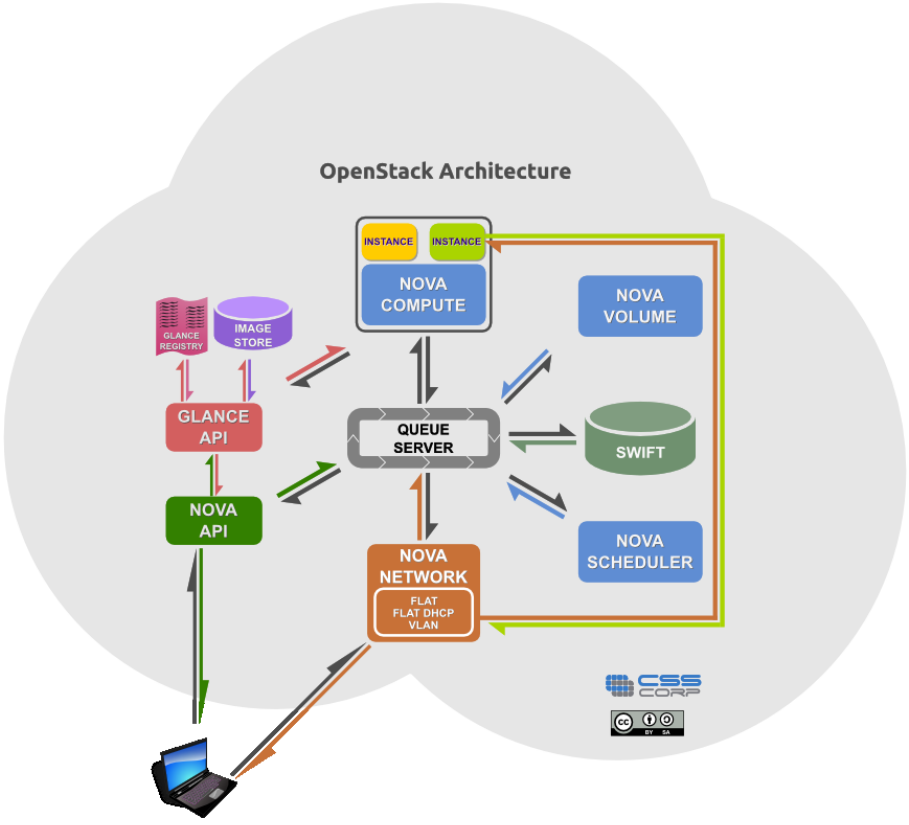

Figure 2.3: Components of OpenStack software diagram
[8]

Authentication and authorization is controlled here for OpenStack services. It is also used to retrieve service catalog of a service in an OpenStack cloud.

- Dashboard (codenamed ”Horizon”)

  Provides web based user interface for OpenStack. It is a simple web GUI to perform operations like controlling instances, assigning IP's and setting access controls etc.

- Block Storage (codenamed ”Cinder”)

  Provides block storage to Guest VM's. This component came from Nova volume service.

- Network and Image Service (codenamed ”Quantum”)

  It provides networking capabilities to a cloud. Users can create their own networks inside and can attach interfaces to them.

- Image (codenamed ”Glance”)

  It provides a catalog and repository for virtual disk images.

2.4 Ubuntu Cloud Infrastructure

Platform plays an important role while building a private or public cloud. Platform should be scalable stable, flexible and built with openness [64].Ubuntu is the most popular platform for cloud computing [57]. The name cloud Infrastructure is used as metaphor name because the structure of the network looks same like a cloud. Ubuntu Cloud Infrastructure is an OpenStack based ready to deploy Infrastructure-as-a-Service (IAAS). [67].Ubuntu cloud infrastructure can be used to deploy both public and private cloud. It is also possible to integrate Ubuntu cloud infrastructure with Amazon Ec2 cloud. It can be deployed locally on one system and on more than one system aswell.

”Canonical, the company behind Ubuntu, was the first company to commercially distribute and support OpenStack, under the Ubuntu Cloud Infrastructure brand” [64]

Ubuntu Cloud Infrastructure is one of the easiest way to deploy a cloud based on openstack. “Ubuntu Cloud Infrastructure is ready to deploy Infrastructure-as-a-Service (IAAS) based on OpenStack. It provides you all the tools you need to offer a private IAAS cloud on your own hardware”[65]. It uses Metal as a service (MAAS) and Juju to setup an openstack based cloud.

Some important points of Ubuntu cloud infrastructure

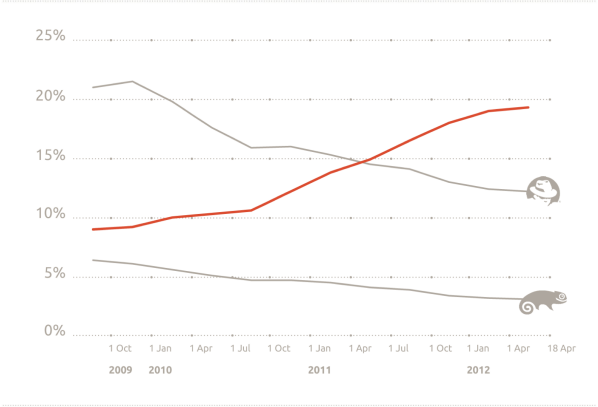

Figure 2.4: Comparing Ubuntu Cloud infrastructure [12]

Ubuntu is the most popular platform for cloud computing [57]

Built into Ubuntu Server, Ubuntu Cloud Infrastructure provides the easiest way to deploy, scale and maintain an OpenStack cloud [42]

Ubuntu Cloud Infrastructure provides the simplest route to deploying an OpenStack Cloud [65]

Ubuntu is the fastest and best-supported route to creating an OpenStack cloud. [64]

According to Matt Ravell, Canonicals MaaS Product Manager, MaaS represents a layer underneath Infrastructure-as-a-Service. The idea is that you use MaaS to orchestrate your hardware, and then you use Juju to orchestrate your applications and workloads. Conceivably, you can use MaaS + Juju to deploy something like OpenStack or Cloud Foundry much faster than if you were manually deploying those instances. [35].

Ubuntu Cloud Infrastructure with MAAS and Juju, can be deployed in three steps[67].

1. Deploy MAAS server
2. Deploy Juju
3. Deploy Ubuntu Cloud Infrastructure with Juju

2.5 MAAS

Setting up a real hardware for a cloud is itself a new chapter. Metal as a Service (MaaS) delivers real servers on demand just like cloud delivers virtual

machines. So upon giving few servers to MAAS, one can save a lot of time because it handles PMI, DHCP, DNS and PXE etc. It supports from a lower scale number of metals (for example 6) to bunch of clusters. It helps in scaling up and down dynamically. It makes it quite easier to set up the hardware to deploy any service, a service that is needed to be scaled up and down according to the demands. [24][63]

Cloud computing is the world of virtualization at front end. But if we go down, it stands on bare metals. And the size of these machines depends on size of cloud. So for a big cloud, the machine's environment should be scaled up but not scaled down. Adding new machines to extend a cloud should be easier. It should be as easy as you connect a plug and play device to your home PC. So here comes the MAAS from canonical to Ubuntu. It can do all this with minimum user intervention.

Canonical is pushing to make Ubuntu the number one OS for running clouds, too. Similarly uncontent with existing descriptors like Platform-as-a-Service (Paas) and Infrastructure-as-a-Service (Iaas), Canonical is announcing their own Metal-as-a-Service (MaaS). [35]

Pxe does not support package management and OS Configurations and MAAS uses cobbler to define profile against each machine. It also supports fantastic features like wake on LAN, to remotely switch on machines. [24]. MAAS can also be used to deploy and prepare systems for lab environments.

MAAS deals with raw metals or machines without installed operating system. MAAS has made it very easy to deploy nodes and these nodes can be updated or retired from the system with a single click. By registering the nodes into MAAS dashboard, it is possible to turn them on and off automatically, depending if the nodes support WOL(wake on lan) features. MAAS works closely with juju to setup any service on a raw bare metal device. It first prepares the machine by installing some basic packages. Later juju takes the machine from MAAS for deploying services.

2.6 Juju and Juju Charms

Juju is a slang word that means magic[16]. It deploys all the required services in a short time. It helps to connect, configure and scale out a service on a cloud with the support of both public and private clouds. Juju has a perfect combination with MAAS, where MAAS deals with all the hardware resources and Juju deals with software services. MAAS and Juju are used together to deploy a complex clusters of servers. Where MAAS is giving power of Metal and Juju charm is helping to connect and configure services on top of these machines.

With over 100 services ready to deploy, Juju enables you to build entire environments in the cloud with only a few commands on public clouds like

Amazon Web Services and HP Cloud, to private clouds built on OpenStack, or raw bare metal via MAAS. Juju manages services, not machines. [55]

Juju configures the machine and it deploys services on these machines. Once the services are started, then relations are created between these services and cloud is ready to be used.

Once MAAS server is deployed then juju is installed on it and then before preparing any other node for deploying open stack component service, it prepares a node for Juju. This node is called bootstrap node. Juju uses this node or system to deploy any service on new nodes.

Juju uses charms to configure and connect the services. Charm store offers 209 different services and each of them can be deployed by a single command [7]. The idea behind charm is to make deployment of services much easier. By writing one single command of Juju, one can deploy a running service like mysql or many others offered by charm store. So rather reading a manual on how to configure a service and trying it, one can use juju charm to configure that service. It is also possible to prepare the nodes and services and then retiring the nodes from Ubuntu cloud infrastructure, so that these nodes or systems can be used for other tasks[61][60].

2.6.1 Systems prepared in Ubuntu Cloud Infrastructure

MAAS deployment needs some resources and that is about ten machines. So a question can arise to some one's mind right away, why just ten machines. The reason is that OpenStack is not built for small clusters but it is for thousands of nodes cluster cloud. Ubuntu cloud infrastructure puts every component of OpenStack and MAAS and juju at separate machines.

Here is a list of services prepared for deploying Ubuntu cloud infrastructure.

- MAAS
- Juju
- mysql
- rabbitmq-server
- keystone
- nova-cloud-controller
- nova-volume
- nova-compute
- glance
- openstack-dashboard

Chapter 3

Approach

This chapter will explain following points

- Objectives to be achieved
- Hardware and Softwares to be used for preparing Environment
- Details of approach and infrastructure design
- Installation and Preparing the Environment

3.1 Objectives

Considering the fact that the introduction and background chapter has explained the basic terms, the objectives mentioned in problem statement will be explained here with some more details.

Ubuntu claims that Ubuntu cloud infrastructure is the simplest route to deploy an OpenStack Cloud[65]. So the question arise whether its deployment in virtual environment is as simple and fast as it is in real environment or not? In the process of building Ubuntu cloud infrastructure, one must build all the required services for OpenStack, as explained in background. Ubuntu has documentation for deploying Ubuntu cloud infrastructure on physical nodes. There could be some challenges while using the same documentation for deploying the nodes in a virtual environment. If there are some challenges then what could be the possible way to encounter these challenges. Implementing these possible solutions helps to fix the challenges or not?

The second objective is to find the current issues in MAAS and Juju while using them in a virtual environment. As explained earlier, MAAS deals with bare raw metals machines, by preparing, installing operating system and doing other necessary configurations. Does MAAS deployment in virtual environment contain some issues? If there are some issues, then what could be the possible solutions to these issues? Implementing the possible solution works or not? Similarly considering all these aspects to find that is it simple to make

Juju up and running for deploying services in a virtual environment? Using the official documentation[66] for MAAS and Juju in a virtual environment lead to some issues or does it work fine? If there are some issues, what are the issues? What could be the possible causes behind them? What could be the possible solutions to these issues? Implementing the possible solution works or not?

A secondary objective is to implement maximum automation for the installation and configuration of the entire system.

3.2 Environment

The practical work will be carried out at Oslo and Akershus University College of Applied Sciences. The hardware resource will only be used for this project in the mean while making sure that it is not used for any other activity, for getting the real results. College has a reliable internet connection for setting up the system.

3.2.1 Physical Server

One powerful physical server will be used during this project. Technical specifications of the server are explained in following table

CUP	*2.4 Ghz*
Number of Processors	*48*
Processor Model	*AMD Opteron(TM) Processor 6234*
Memory	*125 GB*
Hostname	*trident2*

Table 3.1: Physical system specifications

3.3 Approach for Deployment

The experiment shall be carried out on one single physical system. Ubuntu[67] recommends to use Ubuntu 12.04, so Ubuntu 12.04 precise shall be installed on the physical server. A general approach is explained below and is planned before starting the practical implementation of the project. While doing all the practical implementation, challenges and issues shall be taken into the account. The goal is not to build the cloud but is to find the maximum possible

challenges and issues and fixing them, while implementing Ubuntu cloud infrastructure in a virtual environment. Practical investigation of the challenges and issues can lead to the/a little change in the approach.

In order to build a virtual environment KVM hypervisor will be installed and configured. The installation of KVM and its configuration shall be automated by creation of a script. Guest machines require internet connection to install various packages from internet. These machines are supposed to use host machines actual network device to communicate to outside world. This will be done by installing bridge utilities at host system.

3.3.1 MAAS and Juju

The official documentation [67] for preparing Ubuntu cloud infrastructure shall be followed under this case. But if some of the challenges or issues are encountered during the practical implementation then the some other help could be taken from any other available resource. MAAS-DHCP and MAAS-DNS shall be installed on MAAS server to deal with DHCP and DNS services. The general approach will be to first create a virtual network without DHCP and DNS enabled services, followed by creation of a virtual machine for MAAS server.

3.3.2 Infrastructure design

There are various ways to create virtual machines. The virtual machine for MAAS server shall be created by using VMbuilder. An automated network operating system installation shall be carried out. As per Ubuntus recommendation [67]. Ubuntu 12.04 will be installed as an operating system. The process of installing operating system and configuring it for further use shall be automated by creating a script. MAAS and Juju will be installed on the first virtual machine. After installing and configuring other required packages, the new virtual machines shall be created for the sake of deployment of various open stack services. For readers easiness these new virtual machines shall be called nodes in this document. Virsh will be used to create these nodes, such as these machines boots from network and get configured by MAAS automatically. Services required for Ubuntu cloud infrastructure shall be deployed on these nodes, and the relations shall be created between these services.

Figure shows the architecture

Server Technical Environment

The following tables show the basic information of the systems that will be used. Virtual network interface for the virtual machines will be virbr1 and network pv.

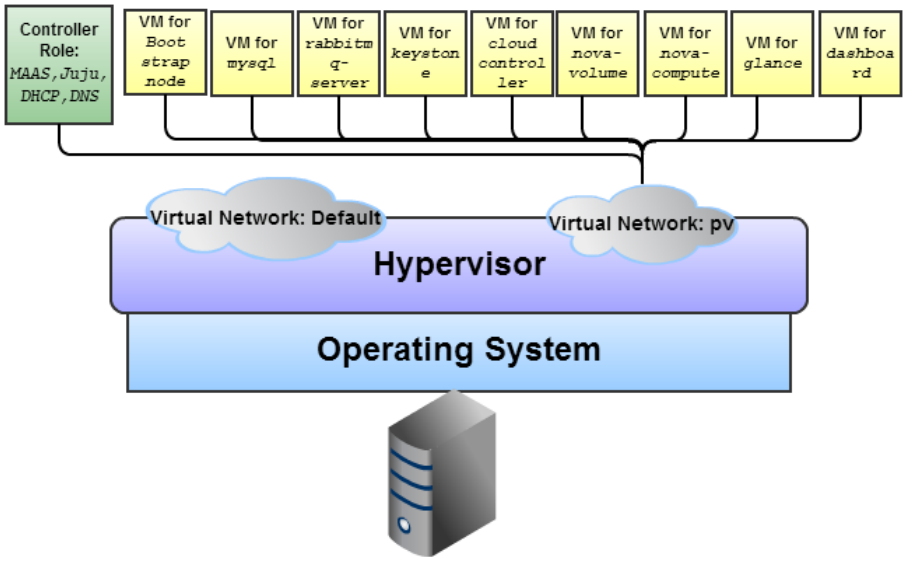

Figure 3.1: Infrastructure design

HOSTNAME	IP ADDRESS	SYSTEM TYPE	ROLE
Trident	192.168.100.1	Real	Host system
Controller	192.168.100.7	Virtual	MAAS and Juju
nodeaabbccddeef0	DHCP	Virtual	Bootstrap node
nodeaabbccddeef1	DHCP	Virtual	Mysql
nodeaabbccddeef2	DHCP	Virtual	rabbitmq-server
nodeaabbccddeef3	DHCP	Virtual	Keystone
nodeaabbccddeef4	DHCP	Virtual	nova-cloud-controller
nodeaabbccddeef5	DHCP	Virtual	nova-volume
nodeaabbccddeef6	DHCP	Virtual	nova-compute
nodeaabbccddeef7	DHCP	Virtual	Glance
nodeaabbccddeef8	DHCP	Virtual	openstack-dashboard

Table 3.2: Server end systems

3.4 Installation and preparation of the environment

This section will explain the practical steps, taken to prepare the environment for the experiment. Ubuntu 12.04 Precise is installed as operating system on the physical system.

3.4.1 Installing KVM and necessary softwares

To install and configure KVM, some help has been taken from the source[62]. The process has been automated by writing a script builder.pl [appendix A].

As explained earlier in section 3.3, to create virtual systems two softwares are used. VM Builder and Virsh has been installed by using some help from source[62][13]. The process of installation has been automated by modifying the script builder.pl[appendix A].

3.4.2 Configuring bridge

To convert the existing network interface of host machine into bridging mode, some bridge utilities has been installed. The network interface file /etc/network/interfaces has been modified. The modified file is as following.

Listing 3.1: /etc/network/interfaces

```
auto lo
iface lo inet loopback
auto em1
iface em1 inet manual
auto br0
iface br0 inet static
        address 128.39.73.54
        netmask 255.255.255.0
        gateway 128.39.73.1
        network 128.39.73.0
        bridge_ports em1
        bridge_stp off
        bridge_fd 0
        bridge_maxwait 0
dns-nameservers 128.39.89.8 128.39.74.66 8.8.8.8 8.8.4.4
```

After configuring bridge, the network services are restarted. All this activity is been scripted in the script builder.pl [appendix A].

3.4.3 Creating virtual Network

Upon installation of KVM, only one virtual network exists for Virtual machines. The network name is default and it has DHCP and DNS services enabled by libvirt. A new network has been created by the name of pv. DHCP and DNS services are not enabled on this new network. As explained earlier in section 3.3, these services shall be managed whether at MAAS server. The network has been created using virsh commands. The created network does natting for traffic engineering between inside and outside network.

Following file is the configuration file of network pv.

Listing 3.2: /var/lib/libvirt/network/pv.xml

```
<network>
  <name>pv</name>
  <uuid>667473e9-544c-ce4e-9eb4-ac09c0c5db4f</uuid>
  <forward mode='nat'/>
  <bridge name='virbr1' stp='on' delay='0' />
  <mac address='52:54:00:AA:CC:E9'/>
  <ip address='192.168.100.1' netmask='255.255.255.0'>
  </ip>
</network>
```

This network has been created automatically by writing a script network.sh [appendix B]. This script is triggered by the script builder.pl [appendix A]. The system is rebooted after the virtual network is created.

3.4.4 Creating virtual machine for MAAS

The first virtual machine is created by using Vmbuilder. Static ip is configured while creating the machine. After conducting network operating system installation the machines sets the password and installs ssh server itself, as this has been told in a script file boot, generated by script creater.sh [appendix B]. Vm builder command is tuned such that, when the operating system starts for the first time it executes that script file boot. It takes approximately fifteen minutes to complete the guest OS installation. Virsh command is used to start the virtual machine, where controller is the hostname of operating system.

A script creater.sh [appendix C] has been prepared to install and configure the above virtual machine. All the above process has been automated by creating the script creater.sh This script does all the steps to create, configure and start the first virtual machine.

`Here are the main operations performed by this creater.sh [appendix C].`

- Creates directory and copying VM-Builder template for the OS to be installed

- Creates VM-Builder partition
- Creates post installation file
  - Setting up OS password
  - Installing openssh server.
  - Creates post installation file
- Runs VM-Builder command to perform OS installation.
  - Sets up IP and network
  - Sets host name and memory
  - Sets operating system user and other variables
- Starting the virtual machine.

Technical specifications of the virtual machine are explained in following table.

Number or Cpus	*5*
RAM	*1024 MB*
Virtual disk memory	*25 GB*
IP	*192.168.100.7(Assigned by DHCP in case scenario 3)*
Hostname	*controller*

Table 3.3: specifications of system "controller"

3.4.5 Creating virtual machine for deploying services

Once MAAS and Juju is installed and configured, the next step will be to create nodes, as explained in section 3.3.1. The nodes will boot and will get IPs from DHCP server. After assigning IPs , operating system on these nodes shall be installed by MAAS server.

Following command is used to create the first node. The value of parameters name, file, and mac shall be changed while creating further nodes.

```
virt-install --name n0 --ram 1024 --arch x86_64 --vcpus 5
--os-type=linux --os-variant=virtio26
 --file=/var/lib/libvirt/images/n0.img --file-size=15
 --graphics vnc --accelerate --hvm --network bridge:virbr1
 --pxe --mac=aa:bb:cc:dd:ee:f0
```

The process of deploying these virtual machines has been automated by creating a nodecreater.pl [appendix H].

Following table shows general details of the nodes.

HOSTNAME	CPUS	RAM	VIRTUAL DISK	ROLE
nodeaabbccddeef0	5	1524	15 GB	Bootstrap node
nodeaabbccddeef1	5	1024	20 GB	Mysql
nodeaabbccddeef2	5	1024	10 GB	rabbitmq-server
nodeaabbccddeef3	5	1024	10 GB	Keystone
nodeaabbccddeef4	5	1024	10 GB	nova-cloud-controller
nodeaabbccddeef5	5	1024	25 GB	nova-volume
nodeaabbccddeef6	5	1024	15 GB	nova-compute
nodeaabbccddeef7	5	1024	15 GB	Glance
nodeaabbccddeef8	5	1024	10 GB	openstack-dashboard

Table 3.4: Nodes specifications

Chapter 4

Results

This chapter describes the achieved by doing this thesis. The first section 4.1, describes the results in detail and also with the practical proves or work done. The initial practical work to setup the environment has already been explained in previous chapter and because of nature of results the remaining practical work has been documented in this chapter.

Explaining this practical work in a separate chapter increases the repetitions of text, because maximum text is covered by the outcomes in the form of results. Also this practical work contains important proves to the results, so the remaining practical work under this chapter. For readers easiness all the achieved results are summarized and listed briefly under section 4.2 till 4.6.

4.1 challenges and issues

As introduced above that this chapter contains some practical work as well. So this section will start by continuing the practical work and with the results.

Installing MAAS

As already described in section 3.3.1, first DHCP and DNS shall be deployed in MAAS server. Official documentation[67] has been followed. The environment has already been prepared and it is stated under section 3.4. A virtual machine has been prepared to deploy MAAS server as guided in section 3.4.4. The installation of MAAS is followed by installing MAAS-DHCP and MAAS-DNS at the same system. After the installation, MAAS user has to be created in order to operate MAAS dashboard.

MAAS as a package is installed on newly created virtual machine, as directed at [58]. Following steps have been performed.

1. To install MAAS

```
sudo apt-get install maas
```

2. MAAS also offers MAAS-DHCP and DNS. Here is a command to install these two services

```
sudo apt-get maas-dhcp maas-dns
```

3. To create a user to log into MAAS web panel, a user has to be created

```
sudo maas createsuperuser
```

4. As MAAS is supposed to Install OS on the nodes, so it needs operating system images. For the first time, the operating system images has to be downloaded manually. Later MAAS checks the new Ubuntu Images automatically, on weekly bases. Command Used:

```
sudo maas-import-pxe-files
```

It takes few minutes to download the images, depends on internet connection speed. MAAS is ready to be logged in via web panel after the downloading process is finished.

```
http://192.168.122.4/MAAS
```

The next step is to create virtual machines, booting from network. As described earlier in background chapter, Ubuntu cloud infrastructure requires ten systems to deploy the cloud. First system MAAS has been prepared now the next virtual machines is needed to be prepared. These machines shall be prepared by MAAS and are called nodes in this document. Hence nine virtual machines or nodes have been created by using virsh command. These nodes are set to boot from network. But the nodes are not turned on yet. MAAS is supposed to turn them on by using its wake on lan feature.

User created above at step 3, is used to log into the dashboard. The next step is to add nodes into MAAS dashboard. There are different ways to add the nodes to MAAS dashboard, explained at [66]. Addition of nodes has been done manually by using the friendly MAAS dashboard.

The MAC address of the already created nodes has to be entered into MAAS dashboard. While adding the nodes, dashboard has option power type that can be selected against each node. Power type can be selected to virtual systems or wake on lan. Apparently this means that MAAS does not support wake on lan for virtual machines. As both wake on lan and virtual systems options cannot be selected at the same time.

4.1.1 Challenge Wake on lan

After adding the nodes into the MAAS dashboard, it should start the nodes for preparation by using its wake on lan feature. But it does not turn on the virtual machines. Both options in MAAS dashboard virtual systems and wake on lan has been tested while adding the nodes. But MAAS does not turn on the nodes itself.

4.1.1.1 Possible reason

The environment is virtual. This is possibly because MAAS does not support wake on Lan for virtual machines.

4.1.1.2 Possible solution

Starting the nodes one by one manually.

4.1.1.3 Implemented solution

Proposed solution works. By turning on the machines manually works.

4.1.1.4 Improving solution

The solution has been improved by using a script doing automation. A script nodecreater.pl [appendix H] has been written for creating and starting the nodes. Now first all the nodes are added into MAAS dashboard and then this script is executed. While creating the machines, it must be make sure that the MAC address matches to the MAC address entered into MAAS dashboard earlier. The guest machines/nodes contact MAAS server and in result MAAS server installs some packages on these nodes and shuts them down.

CONTINUNG DEPLOYMENT

All the virtual machines are started and they boot from pxe to connect to MAAS server for installation. MAAS should prepare these machines.

4.1.2 Challenge Time out

All the virtual machines are started and they boot from pxe to connect to MAAS server for commissioning process, but only two of the machines are entertained at a time by MAAS server. But rest of them ends up with an error message on the terminal no bootable device The machines are contacting MAAS over tftp protocol and are facing time out problem

4.1.2.1 Possible reason

It is a virtual environment or MAAS that does not entertain more than two nodes requests at the same time.

4.1.2.2 Possible solution

Starting the nodes with some time delays

4.1.2.3 Implemented solution

Proposed solution works.

4.1.2.4 Improving solution

The solution has been improved by using a script that does automation. MAAS takes almost 10 seconds to do initial preparation for each node. In order to fix this problem the script nodecreater.pl [appendix H] has been modified by adding sleep of ten seconds between starting each node.

CONTINUNG DEPLOYMENT

MAAS has started to prepare the nodes but each node. MAAS installs some packages on these nodes and shuts them down. The status of the node in MAAS dashboard has changed from commissioning to ready. But operating system is not been installed yet. Operating system should get installed according to the statement.

You can accept and comission the nodes via the web interface. When the nodes have been accepted the selected series of Ubuntu will be installed. [66]

MAAS has installed some packages but not the operating system. Whereas in MAAS dashboard, a new option Start node has appeared, under the node properties. As operating system installation is expected, this did not happen, so selecting start node option should possibly lead to OS installation. But the node is still in a switched off state. MAAS already knows that the nodes are virtual machines but still it does not start the nodes itself. So each node has to be started manually. It is not that difficult job to do for a smaller number of nodes. But for deploying larger number of nodes, it is a time taking process to turn each node on one by one. So a script start.pl [appendix D] has been written to achieve this goal.

Upon starting the nodes, MAAS server starts to install operating system. Each node takes approximately 30 minutes to install the operating system. After the operating system gets installed, the node gets rebooted itself. The state of the node has been changed from Ready to Allocated to root, in MAAS dashboard.

Setting up Juju

The next step is to install and configure juju, as recommended at [67]. Another documentation[61] is followed to install and configure juju as the recommended documentation[67] leads to a broken link[60].

After installing juju, its environment file has to be configured. The environment file contains some basic information for juju, e.g telling juju what environment is used whether its ec2, deploying locally or contacting MAAS for nodes. A sample environment configuration file has been created upon running the command

```
juju bootstrap
```

Environment configuration file

Listing 4.1: /root/.juju/environments.yaml

```
environments:
  sample:
    type: ec2
    control-bucket: juju-c724e86cf1b74e1fabe5c89ed49b72a5
    admin-secret: a387748713b7449ea1c5ca452b7bd88e
    default-series: precise
    ssl-hostname-verification: true
```

But this sample file tells juju to use the systems in ec2. The documentation [1] from Ubuntu does not guide that how to configure a juju environment file to get the nodes from MAAS. The documentation [1] has been used to configure juju environment file, such as juju uses the nodes from MAAS.

The new environment file is as below

Listing 4.2: /root/.juju/environments.yaml

```
environments:
  maas:
    type: maas
    maas-server: 'http://localhost:80/MAAS'
    maas-oauth: 'N7TYzekeU5bfgCP4Dy...s8Bjscj6'
    admin-secret: 'nothing'
    default-series: precise
```

The key of variable maas-oauth has to be changed to the key mentioned in MAAS dashboard. This must be copied from MAAS-dashboard and it can be found under preferences section in MAAS-dashboard.

As juju is going to deploy services on the nodes prepared by MAAS, so juju must have an access to these nodes. The next step is to generate a ssh key, so that each time juju tries to contact new nodes, it should get into the node without any authentication issues. The public part of the key must be copied into MAAS dashboard and then MAAS copies this key into nodes. Later juju can use the nodes without authentication issues.

4.1.3 Issue 409 CONFLICT

After configuring the juju environment file as described above, it has to be bootstrapped so that juju contacts MAAS to get the prepared nodes. And this can be done by using the command

```
juju bootstrap
```

But an error no machines found has been prompt while running this command.

```
Unexpected Error interacting with provider: 409 CONFLICT
2013-05-20 00:35:47,494 ERROR Unexpected Error interacting
 with provider: 409 CONFLICT
```

4.1.3.1 Possible reason

All the nodes were assigned to root user in MAAS. So that is why juju was unable to find any free node.

4.1.3.2 Possible solution

The nodes should not be started second time without using Juju.

4.1.3.3 Implemented solution

So a new installation has been done by using almost the same way but with some changes. Help has been taken from another source[19]. After loging into MAAS dashboard for the first time, a non admin user should be created. A user by the same name must exist at MAAS servers operating system, with root privileges. Nodes in MAAS dashboard should be added by using this non admin user. After adding all the nodes, the state of the nodes in MAAS dashboard is Declared. Before running prepare.pl [appendix D], the nodes must be accepted for commissioning at MAAS dashboard by using the admin user. In result of prepare.pl [appendix D] all the status of the nodes are Ready. But after this step, the nodes should not be started again as they were done. Starting the nodes leads to installation of operating system and assigning the nodes to root user. So juju installation and configuration should be carried out now by using the newly created user at the operating system. The username is adnan and it has root privileges. As explained earlier, it is important that the username should match with the non-admin user in the MAAS dashboard.

After juju is configured, as explained in previous attempt, bootstrapping the environment leads to find the nodes for juju. Juju bootstrap command is ended up with a successful message. All these operations should be performed by using user adnan.

CONTINUNG DEPLOYMENT

Now MAAS dashboard shows a status change against first node. The status has changed from Ready to Allocated to adnan. Virsh command is used to start the node, where n0 is the name of the host, configured while creating the node prepare.pl [appendix E]

```
Virsh start n0
```

MAAS took approximately thirty minutes to perform operating system installation on this node. The node gets restarted and stops at the login screen.

But as no password is configured but only the keys, so one must wait until MAAS copies the ssh keys to this node.

After the keys are copied to the node, the node is ready to be used by juju.

4.1.4 Issue: Juju status

Juju status command has been executed to see the status. It tries to find bootstrap node but fails to do so.

4.1.4.1 Possible reason

It seems that juju has some problems in finding the first prepared node. Making an entry in /etc/hosts file by writing the first prepared node hostname with its IP, fixes this issue and juju status works. This means there is an issue problem in MAAS-DNS configuration.

4.1.4.2 Possible solution

1. Debugging the MAAS-DNS
2. Making an entry in /etc/hosts file by writing the first prepared node hostname with its IP

4.1.4.3 Implemented solution

The second solution Making an entry in /etc/hosts file by writing the first prepared node hostname with its IP, fixes the issue. As the DNS service does not seems to be in working state. After debugging, it is found that MAAS-DNS is not working fine, since the nodes are unable to resolve the addresses. As the documentation[67] does not explicitly ask to configure MAAS-DNS, so it assumed that MAAS-DNS should work under auto configuration. Moreover no documentation has been found on MAAS-DNS. Various attempts have been made to fix this issue but it is still facing the same issue. While trying to reinstall MAAS-DNS an error occurred.

```
unable to find package MAAS-DNS.
```

It is discovered that MAAS-DNS was removed from the main repository and it is obsoleted. So, some changes are made in the approach for following reasons.

- MAAS-DNS was giving problems
- No documentation found on configuring MAAS-DNS
- MAAS-DNS is removed from the main repository

- Another documentation[19] by MAAS team has been found, which is using dnsmasq to deal with DHCP and DNS issues. This documentation is for deploying MAAS and Juju in virtual environment.

CONTINUNG DEPLOYMENT

The documentation[19] by MAAS team directs about deploying MAAS in a virtual environment.

After installing MAAS package, the next step is to install dnsmasq, as recommended [19].dnsmasq covers both DHCP and DNS.

The same steps needed to be followed as already done in previous installations but with an exception of installing dnsmasq instead of MAAS-DNS and MAAS-DHCP. Cobbler has been configured as explained in the document[19].

4.1.5 Issue: Cobbler Errors

An error is prompt upon adding the first node into MAAS dashboard

```
Error: Unknown cobbler problem
```

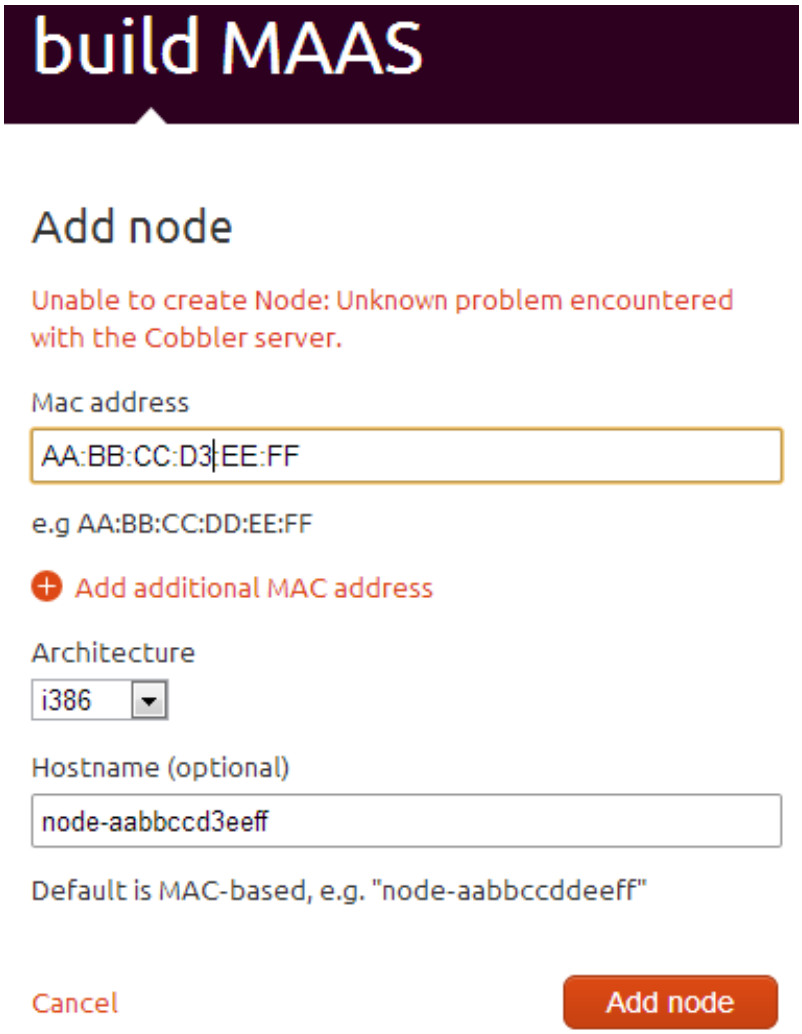

Figure 4.1: Unknown cobbler error.

The error stayed on the screen just for 2 seconds and vanished. The node was added into MAAS-dashboard. While trying to add a new node, another error has been appeared

```
unable to create Node: Timed out.
```

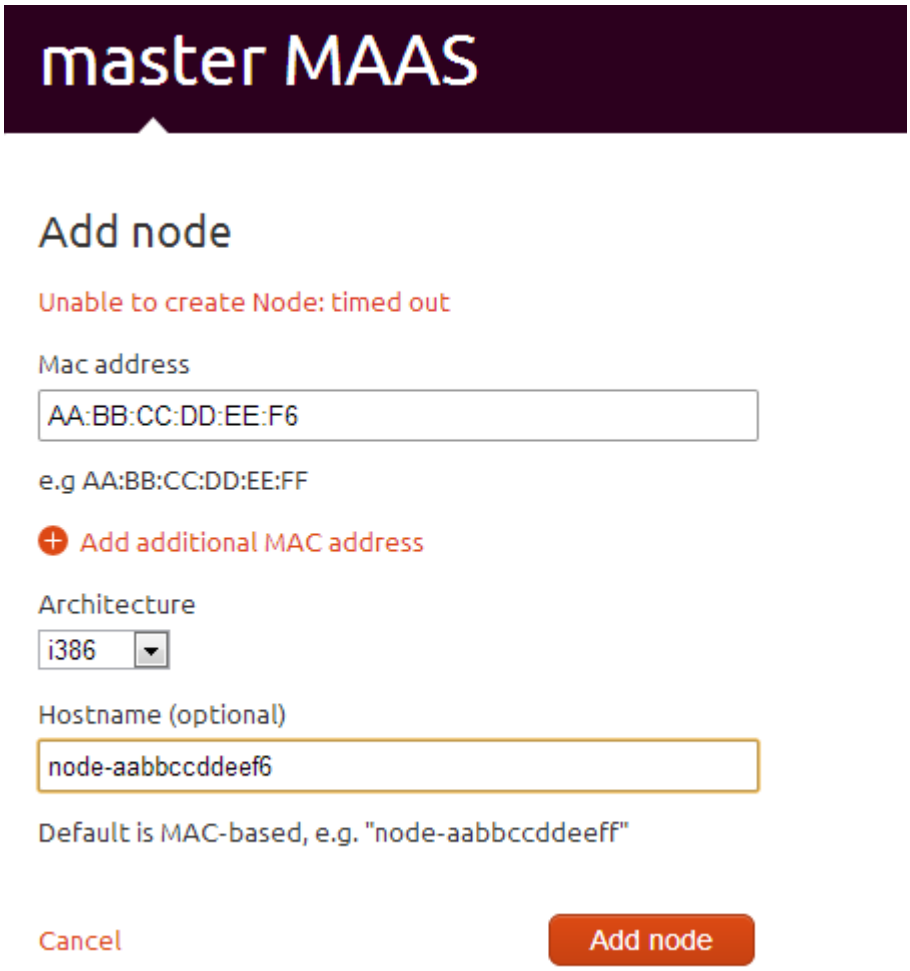

Figure 4.2: cobbler timed out error

4.1.5.1 Possible reason

After some investigating cobbler log, it is discovered that the problem is with MAAS and dnsmasq.

4.1.5.2 Possible solution

As per the solution recommended at [10]

This is caused by a very long delay by dnsmasq when it gets restarted after node additions. Edit the PSERV_TIMEOUT value in the Django settings.py so it's 30 instead of 7.

4.1.5.3 Implemented solution

The recommended solution works, if the value of PSERV_TIMEOUT is raised to 60. Though the error is gone but one must wait for one minute after adding

each node into MAAS dashboard. Also, while accepting the node for commissioning, later while running juju bootstrap command, checking status. One must wait for one minute at every step against each node. Waiting means, system takes one minute to complete your query.

But this has fixed Juju status problem. Now Juju status command works fine and DNS service is up and running.

4.1.5.4 Improving solution

Considering the cobbler error in previous section there is another solution found at [6]

This is fixed in trunk with the removal of Cobbler[6]

So this means MAAS-DHCP should be installed to deal with DHCP issues and dnsmasq should be configured for DNS services.

Installation has been carried out as directed at [19]. But this time cobbler is not configured for DHCP but MAAS-DHCP is used for this DHCP services. To avoid any error, previously encountered issues and challenges are kept into the account while installation has been made so that the same encountered issues and challenges may not occur again.

This solution worked fine.

CONTINUNG DEPLOYMENT

Command Juju status shows that juju has prepared the node. This first prepared node is called bootstrap node. Bootstrap node is used by juju to deploy services on other the other nodes.

Deploying Charms

The official documentation[67] is used to carry out this procedure. First step is to deploy mysql service by using following command.

```
Juju deploy mysql
```

As concluded from the documentation, Juju should contact MAAS server to get one more node to deploy mysql. After running above command, one can check the instance ID by running juju status command. Juju status should tell the instance ID of the new machine that it got for MAAS for deploying mysql service. Instance ID is the term used by juju for hostname of the node.

4.1.6 Issue Pending ID

After running the juju deploy command, though juju status command shows another node but with a pending instance ID. Which apparently means that juju is still in the process of getting the node from MAAS.

4.1.6.1 Possible reason

Because in the environment file, the address of MAAS node is configured as 'http://localhost:80/MAAS'.

4.1.6.2 Possible solution

So by making some changes in environment file, it might solve this issue. Hence localhost is replaced by the IP address of the MAAS server, as stated below. This works by putting hostname of the server as well.

```
maas-server: 'http://192.168.122.4:80/MAAS'
```

4.1.6.3 Implemented solution

This possible solution described works.

CONTINUNG DEPLOYMENT

There was more than one pending instance IDs in result of command juju status, because of debugging process to fix this issue. Different tries has been made to deploy various services, during the debugging process in previous section. So a clean installation is required in order to avoid any surprise problems. A new installation has been carried out as it was done in the previous section but with some changes. The change has been made only in juju environment file as explained above.

4.1.7 Issue fault code 99

While doing a new installation above, a new issue has been encountered. Previously explained procedure has been followed for adding the nodes but MAAS dashboard shows an error while adding few of the nodes. For example a first node gets added to MAAS dashboard without any error but then upon adding few more nodes it gives error against few of them. The nodes get added into MAAS dashboard but the error is popped afterwards. Just to make clear for the reader, the virtual machines are not started yet against these nodes, as the previously explained procedure is fallowed.

The error is as fallows

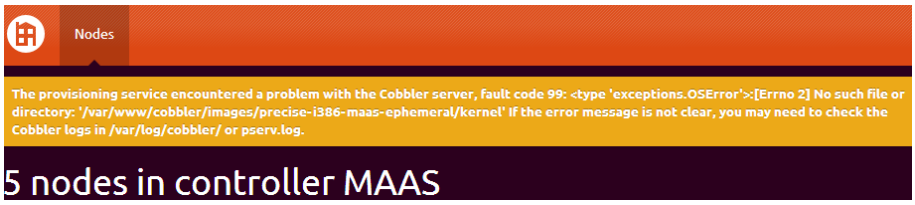

Figure 4.3: Mass dashboard error

4.1.7.1 Possible reason

Fast addition of nodes in MAAS dashboard.

4.1.7.2 Fixing fault code 99 issue and continuing

The log files have been explored to fix this issue. When a node is added into the MAAS dashboard, some operations are performed at backend which includes: cobbler defines a profile against each new machine, copying images, generating GPXE/PXE configuration files and running shell scripts to triggers some changes.

4.1.7.3 Possible solution

Waiting for some time between adding nodes into MAAS dashboard might fix this issue.

4.1.7.4 Implemented solution

It is discovered that given a small pause between adding the node fixes this issue. One should not be too quick in adding the nodes into MAAS dashboard.

CONTINUNG DEPLOYMENT

Deploying Charms

Command to deploy mysql service

```
Juju deploy mysql
```

Status of juju now shows the hostname of the node assigned to mysql. The other way to check the assigned nodes hostname is by using MAAS dashboard. The assigned node has been started using virsh command. The next step is service to deploy is

```
rabbitmq-server.
```

this service has been deployed in the same manner as mysql service was deployed. The assigned node has been started.

Create a configuration file

The services can be deployed by using some pre-defined configurations. As recommended at [67], a file has to be created, stating the configuration required. Such file has been written and is saved by the name openstack.cfg.

Listing 4.3: /adnan/openstack.cfg

```
keystone:
    admin-password: "openstack"
nova-cloud-controller:
    network-manager: "FlatDHCPManager"
nova-volume:
    block-device: "xvdb"
    overwrite: "true"
```

The file is used to set the password for keystone service. Similarly while deploying nova-cloud-controller it should use the specified network manager. Block storage name is mentioned for nova-volume.

```
To deploy these services .following commands have been executed.
juju deploy --config=openstack.cfg keystone
juju deploy --config=openstack.cfg nova-cloud-controller
```

The assigned nodes have been started.

4.1.8 Issue: block mapping

To deploy nova-volume following command is used

```
juju deploy --config=openstack.cfg nova-volume
```

But this gives an **error**

74 INFO Searching for charm cs:precise/nova-volume in charm store 2013-03-22 19:53:32,801 ERROR while parsing a block mapping in "", line 1, column 1 did not find expected key in "", line 13, column 5

4.1.8.1 Possible reason

Some study about nova-volume charm[54] tells that its expecting a hardware device by the name of xvdb. But not such separate disk drive by the name of xvdb is attached.

4.1.8.2 Possible solution

Adding a virtual drive

4.1.8.3 Implemented solution

This can be done by adding a virtual block device.

```
block-device: "/var/lib/nova-volumes/vol1.img|20G"
```

It should solve this issue. But juju is still giving the same error.

4.1.8.4 Improving solution

After debugging, it is found that juju is picky with the number of spaces to be used in openstack.cfg. Hence extreme care must be taken while writing openstack.cfg. The number of character spaces before block-device line must be equal to four.

The nova-volume section in openstack.cfg is updated.

Listing 4.4: /adnan/openstack.cfg

```
keystone:
    admin-password: "openstack"
nova-cloud-controller:
    network-manager: "FlatDHCPManager"
nova-volume:
    block-device: "/var/lib/nova-volumes/vol1.img|20G"
    block-device: "xvdb"
    overwrite: "true"
```

After putting the spaces carefully, the command worked.

CONTINUNG DEPLOYMENT

And then the remaining services are deployed by using following commands.

```
juju deploy nova-compute
juju deploy glance
juju deploy openstack-dashboard
```

The assigned nodes have been started. Juju status shows that all the services have been deployed successfully.

4.1.9 Challenge booting nodes

Juju deploy has been executed and all the nodes were started. Installation of the operating system is completed and the nodes got rebooted. Now MAAS should copy the ssh key to the node. The nodes start up and boot from local hard disk. After downloading some packages from internet, the node hangs. All the nodes were turned off then started back

```
Virshdestroy n1
Virshstart n1
```

But each node stops at the grub loader and asks for a human interaction to select an operating system to boot. Default option was selected. Now MAAS has copied the ssh keys to all the nodes.

4.1.9.1 Possible reason

Deploying many services at the same time at different nodes is possible the reason.

4.1.9.2 Possible solution

Making some delays between prearing the nodes. Such that once a node is prepared then the next node should be started to be prepared.

4.1.9.3 Implemented solution

A same process is repeated as done in previous section but with an exception of not starting the nodes after executing the juju deploy commands. Starting the nodes one by one, by making sure that previous node has operating system got installed.

4.1.9.4 Improving solution

To encounter this challenge a delay of 30 minutes has been made while starting each node. The script start.pl [appendix D] has been modified such that there is delay of 30 between starting each node.

CONTINUNG DEPLOYMENT

Nodes were prepared without any error. The next step is to build relations between these nodes. It has been tested that node can ping each other by using the hostnames.

Establishing relation between services

The next step is to create relations between these services as recommended at [67]. Following commands are used to create relations

```
juju add-relation keystone mysql
juju add-relation nova-cloud-controller mysql
juju add-relation nova-cloud-controller rabbitmq-server
juju add-relation nova-cloud-controller glance
juju add-relation nova-cloud-controller keystone
juju add-relation nova-volume nova-cloud-controller
```

```
juju add-relation nova-volume mysql
juju add-relation nova-volume rabbitmq-server
juju add-relation nova-compute mysql
juju add-relation nova-compute rabbitmq-server
juju add-relation nova-compute glance
juju add-relation nova-compute nova-cloud-controller
juju add-relation glance mysql
juju add-relation glance keystone
juju add-relation openstack-dashboard keystone
```

ISSUE

Juju status shows relation has been created successfully between all the nodes.

Exposing Services

The next step is to expose dashboard and cloud control services.

```
juju expose openstack-dashboard
juju expose nova-cloud-controller
```

As per the official documentation: At this point, the Openstack cloud has been deployed and should be functioning [67]. Openstack dashboard can be accessed by accessing public address of OpenStack-dashboard node.

```
http://node-aabbccddeef8.localdomain/horizon
```

Password was configured in openstack.cfg file, while deploying keystone. It should work by using following credentials

```
username admin
password openstack
```

4.1.10 Issue: Authentication

Upon punching in the credentials at OpenStack dashboard, it prompts an error.

```
Error: An error occurred authenticating. Please try again later.
```

4.1.10.1 Possible reason

If it is about wrong username or password than there should be an authentication error but not an error stating please try again later. Juju status does not report any relation error between dashboard and keystone. But now after a minute juju states a relation error between compute-node and nova cloud controller node.

These two issues are possibly not related to each other because keystone deals with the authentication issues. To make sure that novastack-dashboard is communicating to keystone, some investigation is required. Network traffic has been monitored at keystone node to see if there are some incoming packets, upon a login request at dashboard. This apparently shows that the problem is at keystone server, because keystone is receiving the request. After digging more into keystone, its discovered that there is no file: "keystone.passwd", under the directory "/var/lib/keystone/". And this file should have the password from openstack.cfg.

And the error reported by juju is a relation error between nova-cloud-controller and nova-compute. Openstack.cfg was used while deploying nova-cloud-controller. There were some issues while deploying nova-volumme by using openstack.cfg. So its possible that there are some issues while deploying a node by using preconfigured options like openstack.cfg.

Keystone generates a random password for itself, if no pre-installation password option is provided. So by creating a new keystone without using openstack.cfg, might fix logging issue. Furthermore it is possible that while deploying nova-cloud controller by using openstack.cfg there were some issues. So perhaps nova-cloud controller can be prepared without using openstack.cfg. Network manager can be configured later after nova-cloud controller is up and running.

4.1.10.2 Possible solution

So the possible solution could be to re-create keystone, volume and nova-cloud-controller without using openstack.cfg.

4.1.10.3 Implemented solution

Three new nodes are prepared and the old nodes are to deploy nova-volume, nova-cloud-controller and keystone. Previously used nodes against these services were terminated.

CONTINUNG DEPLOYMENT

Relations have been created again. . Keystone has generated a random password under the directory /var/lib/keystone/. So now it is possible to login to OpenStack dashboard.

4.1.11 Issue: Internal Server Error

All the options in open stack dashboard do not work. E.g upon clicking the sections Instances and volumes, Images and Snapshots, access and Security under the tab project, It generates an error:

```
Internal Server Error
An unexpected error occurred while processing your request.
Please try your request again.
```

4.1.11.1 Possible reason

Nova cloud controller is not told to use Flat-DHCP for networking.

4.1.11.2 Possible solution

This[18] recommends that by placing openstack.cfg under the directory .juju, should fix this issue. Moreover juju creates the environment file under the same directory.

A new installation should be carried out because allot of changes were made during the debugging process. The installation will be done in the same fashion as it was carried out before, but with an exception of changing the file path. Openstack.cfg is to the directory .juju.

Listing 4.5: /adnan/.juju/openstack.cfg

```
keystone:
    admin-password: "openstack"
nova-cloud-controller:
    network-manager: "FlatDHCPManager"
nova-volume:
    block-device: "xvdb"
    overwrite: "true"
```

CONTINUNG DEPLOYMENT

Deployment has been started from start.

4.1.12 Issue: MAAS pxe boot issue

Now upon starting over from building MAAS a new Issue has been encountered. MAAS server is not responding to Pxe boot clients. When the nodes are booted for the first time, they are supposed to contact MAAS for installation. DHCP server directs them to MAAS server but MAAS is not replying to these nodes. No changes have been made in the installation procedure.

4.1.12.1 Possible reason

tftp package not found at MAAS.

4.1.12.2 Possible solution

It has been discovered that MAAS does not contain any service to respond to pxe clients. This can be fixed by adding such features

4.1.12.3 Implemented solution

Installing some extra packages. Commands used:

```
sudo apt-get install maas-enlist tftpd-hpa
sudo maas-import-isos -u
```

This has fixed the issue and now the nodes are able to boot from MAAS.

CONTINUNG DEPLOYMENT

Nodes were deployed by placing openstack.cfg in the right directory. Now OpenStack dashboard works with the predefined password. Which possible means that, juju has configured the nodes according to the contents provided in openstack.cfg file. So nova cloud controller is installed with the network configuration set to Flat DHCP Manager.

4.1.13 Issue: relation error

Juju status reports the relation error between nova-cloud-controller and nova-compute. Juju debug-log command has been executed to fix this issue but that does not work either.

4.1.13.1 Possible reason

From juju logs it seems that nova-compute is having some DNS problems.

4.1.13.2 Possible solution

Using the possible solution suggested at: 4.1.4.2 Possible solution, Making an entry in /etc/hosts file by writing the first prepared node hostname with its IP

4.1.13.3 Implemented solution

Implementation has been done in appendix section Approach1Appendix F. Using domain name resolution locally at each system by using /etc/hosts files. Approach1Appendix F explains the possible solution and its implementation in detail. A summary of the scenario is explained here:

The system got deployed successfully by managing the DNS address translation at KVM and at each local system. KVM libvirt does the basic address translation. But it does not recognize the domain names if they are not created in libvirt. So entries in /etc/hosts was made at every virtual machine to make sure that the system can access each other using the names with the extra string attached .localdomain.

The same relation error occurred again

4.1.14 Issue: Juju charm

Nova compute charm is dependent on Domain service.

4.1.14.1 Possible reason

As in the previous section it is discovered that nova-compute charm is written in such a way that it contacts the domain server to use DNS service. It does not work locally using /etc/hosts.

4.1.14.2 Possible solution

Building a separate domain controller first and then creating MAAS server and registering MAAS server. Approach2Appendix F explains the possible solution in detail.

4.1.14.3 Implemented solution

Possible solution has been implemented with automaton. Step by step implementation in section: Approach2Appendix F.

During the experiment a challenge was encountered

4.1.15 Challenge: DHCP not found

In the previous section, nodes were unable to find DHCP server. The nodes first booted and then got the IP from MAAS server and got prepared by MAAS. But the next time when operating system installation started not were unable to find DHCP service in the network. Approach2Appendix F explains it more in detail.

4.1.15.1 Possible reason

Dnsmasqs DHCP does not support default KVM network card models.

4.1.15.2 Possible solution

Changing the network card model of node.

4.1.15.3 Implemented solution

Model change to rtl8139 fixes this problem. Automation implemented. Approach2Appendix F explains it more in detail.

4.1.16 Issue: Fixing Juju charm

Same relation error between nova compute and nova cloud controller. So now the issue has become: whether it is issue nova compute charm or with MAAS and Juju configuration.

4.1.16.1 Possible reason

In previous approach, to fix the issue it has been discovered that Juju is not assigning the created domain names into the public address it assigns to the machines. But it attaches .localdomain with each of the node name.

4.1.16.2 Possible solution

Creating domain controller and deploying DHCP services by not following official documentation. Most importantly keeping the name of the domain to localdomain. Deploying MAAS and Juju on the same system as that of domain controller.

4.1.16.3 Implemented solution

Automation has also been implemented. A script resolver.pl[appendix I] creates the domain by the name of localdomain and setups of DHCP service on MAAS server. After setting up domain and DCHP service on MAAS server MAAS and juju has been deployed.

This fixes this issue. The relation error between nova cloud controller and compute is fixed. Ubuntu cloud controller has been deployed.

4.2 Challenges summary

Challenge found	Possible Reason	Possible solution	Implement	Status	Automation
wake on lan	MAAS no wake on support lan for virtual systems	**a)** Turning on these nodes **b)** Automation	✓ ✓	✓ ✓	N-A ✓
Time out	MAAS does not support handling more than two requests at a time	**a)** Time delay between turning on the machine manually **b)** Automating time delay	✓ ✓	✓ ✓	N-A ✓
booting nodes	MAAS OS install at more than three nodes at the same time	**a)** Manual time delay between OS installation **b)** Automating time delay	✓ ✓	✓ ✓	N-A ✓
DHCP not found	DHCP does not support default interface model	Model change to rtl8139	✓	✓	✓

4.3 Issues summary

4.4 Automation

This section shows a brief summary of created scripts.

Issue found	Possible Reason	Possible solution	Implement	Status	Automation
409 CONFLICT	Nodes assigned to MAAS root user	Not starting the nodes Configuring Juju first	✓	✓	N-A
Juju status	MAAS-DNS	a) /etc/hosts b) dnsmasq	✓ ✓	✓ ✓	N-A
Cobbler Error	Long delay	a) PSERV_TIMEOUT increase b) dnsmasq	✓ ✓	✓ ✓	N-A
pending ID	Juju Locally search	Modifying Juju ENV config	✓	✓	✓
fault code 99	Cobbler profiling	Time delay between adding the nodes in MAAS-Dashboard	✓	✓	N-A
block mapping	a) Virtual drive a) number of spaces	Modifying Juju pre-config and inserting spaces	✓ ✓	✓	✓
Authentication	Password file missing at keystone	Generating random password	✓	✓	N-A
Internal Server Error	Flat DHCP	Changing openstack.cfg path	✓	Error	N-A
MAAS pxe boot issue	tftpd service	Installing services	✓	✓	✓
Relation error	MAAS-DNS	/etc/hosts Creating Domain controller	✓	Error	✓
JujuCharm	Defining Domain	Implemented by creating separate Domain controller with DHCP	✓	Error	✓
localdomain	Juju configuration of	Defining domain name locally on	✓	✓	✓

4.4.1 Automated virtual environment for Ubuntu cloud Infrastructure

This script builder.pl [appendix A] does all the tasks to build an automated environment for deploying Ubuntu cloud infrastructure.

`Here are the main operations performed by this builder.pl [appendix A].`

- Installation of all the required packages
- configuring the packages
- configuring Bridge
- Configuring Virtual networks
  - setting up a new virtual network without DHCP and DNS services enabled
  - Changing the existing virtual network by adding pxe support and address to MAAS server.
- Rebooting the host machine.

4.4.2 Automated virtual machine for MAAS

This script creater.sh [appendix C] does all the tasks to build an automated virtual machine for deploying MAAS.

`Here are the main operations performed by this creater.sh [appendix B]`

- Creates directory and copying VM-Builder template for the OS to be installed
- Creates VM-Builder partition
- Creates post installation file
  - Setting up OS password
  - Installing openssh server.
  - Creates post installation file
- Runs VM-Builder command to perform OS installation.
  - Sets up IP and network
  - Sets host name and memory
  - Sets operating system user and other variables
- Applying firewall rules at host machine for MAAS traffic.
- Starts the prepared virtual machine.
- Copying the files and scripts into the virtual machine

4.4.3 Deploying MAAS and Juju and fixing some Issues

`Here are the main operations performed by this maas1.sh[Appendix G]`

- installing some MAAS packages
- Importing Operating system images
- Installing Juju and Charm tools.
- Setting up juju environment file.
- Setting up OpenStack configuration file openstack.cfg
- Rebooting Machine
- Fixing Issues
  - pending ID
  - block mapping
  - MAAS pxe boot issue

4.4.4 Creating VM for services deployment and fixing challenges and issues

`Here are the main operations performed by this nodecreater.pl[Appendix H]`

- Creating Virtual machines for service deployment
- Fixing Challenges
  - Time out
  - DHCP not found

4.4.5 Automats juju services deployment

`Here are the main operations performed by this juj.pl [App J]`

- Deploying OpenStack services
- Connecting OpenStack Services
- Exposing OpenStack dashboard

4.4.6 Automats Starting nodes and fixing some challenges and issues

`Here are the main operations performed by this start.pl [App D]`

- Starting the nodes
- Fixing Challenges
  - booting nodes

4.4.7 Creating Domain Controller and DHCP and fixes some challenges and issues

`Here are the main operations performed by this resolver.pl[Appendix I]`

- Installing DHCP and DNS packages
- Configuring Domain Controller and DHCP Controller
- Configuring pxe boot requests
- Fixing Issues
  - Relation error
  - JujuCharm
  - localdomain

Chapter 5

Analysis and Discussion

This chapter treats analysis and discussions of all the work done. Each result is first analyzed and then discussed. For the sake of readers easy understating, firstly analysis and discussions have been made on automation script that prepares the environment for deployment. And the all the challenges and issues were discussed. After discussing challenges and issues the remaining automation scripts have been discussed that helps to do remaining automation. The issues fixed by doing automation or writing scripts have been analyzed and discussed in this section as well.

Various challenges and issues were encountered during the deployment it in a virtual environment. Each challenge and issue was investigated to find the possible reason of its occurrence and also to find the possible solution to fix it. Furthermore the suggested possible solutions were implemented to fix these challenges and issues. Scripts were produced to make it easier to deploy such a cloud in a virtual environment and also to avoid maximum challenges and issues while the deployment process.

5.1 Automation

This section does analysis and discuss of the initial scripts. The initial scripts prepare a fully automated environment so that Ubuntu cloud may be than deployed on the environment. Note that the remaining scripts shall be discussed after analyzing and discussing challenges and issues.

5.1.1 Preparing environment

Virtual environment could have been prepared manually to but automation approach was chosen to make automated creation of environment. The script ”builder.pl” [appendix A] was produced to make an easy automated deployment of the environment. Script takes care of everything to setup the environment to run a virtual machine. It does package installation, configuring bridge over host machine, and building virtual network. It calls another script network.sh [appendix B] to create virtual network. Where network.sh [appendix B] replaces the does some changes in the existing virtual network default and

it also creates another network called pv. It is possible to make one script out of these two but because of time limitations and putting more focus on Ubuntu cloud infrastructure, this was not done. Otherwise it is easily possible by copying all the code from network.sh [appendix B] to builder.pl and modifying it according to perl syntax, or the other way around.

The script "builder.pl" [appendix A] is good enough to achieve automation but a weakness of this script is that it does not modify the existing network default. But it destroys the existing network and creates a new network by the same name. This does not create any problem but it is not considered a professional approach. It is possible to remove this weakness easily by using e.g some perl options. Keeping automation as secondary objective and time limitation did not allowed doing that.

So script "builder.pl" [appendix A] installs and configures everything and reboots the system. The system becomes into ready state for deploying virtual machines.

5.1.2 Virtual machine deployment

The next step was to create virtual machines for deploying MAAS. This could have been done manually but this was also automated by writing another script creater.sh[Appendix C] It does all the required tasks from creating machine, setting the password and installing basic services like opens-ssh. It uses VM builder command to create machine and install operating system from network. After creating the virtual machine it starts the virtual machine. Upon running the machine for the first time a post installation script gets executed automatically. Which sets the password and installs ssh services. The script creater.sh[Appendix C] also sets the firewall rules for forwarding the http traffic, coming to host machine, to this machine. So that it becomes possible to access MAAS dashboard using live IP, assigned to host machine. It copies some scripts and files into this newly created virtual machine for MAAS. The script does a good job to achieve automation for building virtual machine.

5.1.3 Installing and setting up MAAS and Juju

MAAS Service is installed by running a command manually. This could have been automated by using some perl functions like expect, to deal with the graphical prompts, during installation of MAAS. Once MAAS package is installed, the script maas1.sh[Appendix G] is executed. This script does the remaining installation of MAAS by installing some other packages. Furthermore it imports MAAS operating system images. It installs and configures the environment for juju. It further also generates openstack.cfg file. Finally it restarts the system. The script does all the required automation but it can be improved by doing slight modification into the syntax. The environment and openstack configuration files are copied because juju is very choosy with respect to the number of spaces and quotes. So the script can be improved by generating new files, by just printing these files using perl, rather copying them.

After the system gets rebooted, nodes are added manually into MAAS-dashboard. The nodes are accepted for commissioning manually one by one in MAAS dashboard. MAAS-CLI tools and commands [53] can be used to accept all the nodes by running a single command, but in that case any unwanted system can gets registered under MAAS. It clearly means MAAS will accept and commission all the systems connecting to MAAS. But the systems can be accepted by describing their ids using maas cli tools. So rather accepting the nodes manually, the system ids can be used to accept these nodes via command line. This could be a good way to improve automation by setting up MAAS cli tools. Because the step by step official documentations[67][19] were followed for basic deployment to test the quality of documentation, so MAAS-cli tools were not used.

Ssh keys was generated and pasted into MAAS-dashboard manually. Generating them via script was possible but pasting them into maas dashboard is little difficult via script. MAAS or Juju configuration files are needed to be explored in order to find that where MAAS puts this ssh-keys. Because the step by step official documentations[67][19] were followed for basic deployment, so this was not implemented in automation process.

5.2 Challenges, issues

5.2.1 Challenge: Wake on lan

Once a machine is registered inside MAAS dashboard, the machine should get started by MAAS automatically. MAAS sends wake on lan call to these machines to turn them up and it prepares the machines set to network boot. This scenario does not work in virtual environment. It seems from the options MAAS has in its dashboard that MAAS does not support this features for virtual machines. Because while adding a node only one option from the two given node type options can be selected. These options are virtual systems and wake on lan. So this apparently means that MAAS does not support wake on lan feature for virtual machines. Both options were tested during the experiments but they did not help to start the virtual machines over wake on lan call. Starting virtual machines manually fixes makes it possible to start preparing the nodes. But starting the one by one manually is a time consuming process.

Hence a solution was implemented in a script which creates and starts the virtual machines. This was done by writing a script nodecreater.pl[Appendix H] . It uses simple virsh commands to create and boot the machines from network. It also marks an entry in a text file after setting up a virtual machine. It does all the required automation but can be improved by using some variables to decrease the line of code. So first all the virtual machines, also called nodes, were added into MAAS dashboard and then this script was executed so that it may create and start these nodes. Starting the nodes via script may not be the best solution. Other solutions could be by exploring the other end (virtual machines) but not MAAS. Investigating how virtual machines can be started

by lan call. And if applicable then it should also be investigated that this might cause some resource consumptions, while the virtual machines are in off state.

5.2.2 Challenge: Timed Out

All the nodes were started using the script nodecreater.pl[Appendix H] .But MAAS accepted only two nodes at the same time for communing. The other nodes ended up with the time out error. It is a virtual environment or MAAS that does not entertain more than two nodes requests at the same time. This was checked by assigning more system resources to MAAS but did not help. So the previously written script nodecreater.pl [Appendix H] was modified by inserting time delay of 10 seconds, between starting each node. MAAS uses cobbler to deal with pxe boot issues. So this can be further investigated to find a batter possible solution.

5.2.3 ignored IPMI error

When MAAS started to prepare the nodes, each node shows an error at its terminal screen, right before it shuts down.

```
Setting up freeipmi-tools (0.8.12-3ubuntu1) ...
Processing triggers for libc-bin ...
ldconfig deferred processing now taking place
FATAL: Error inserting ipmi_si (/lib/modules/3.2.0-31-generic/kernel/drivers/cha
r/ipmi/ipmi_si.ko): No such device
Traceback (most recent call last):
  File "/tmp/part-001.NMZaTk/bin/maas-ipmi-autodetect", line 98, in <module>
    main()
  File "/tmp/part-001.NMZaTk/bin/maas-ipmi-autodetect", line 93, in main
    IPMI_IP_ADDRESS = get_ipmi_ip_address()
  File "/tmp/part-001.NMZaTk/bin/maas-ipmi-autodetect", line 37, in get_ipmi_ip_
address
    return res.group()
AttributeError: 'NoneType' object has no attribute 'group'
Traceback (most recent call last):
  File "/tmp/part-001.NMZaTk/bin/maas-signal", line 262, in <module>
    main()
  File "/tmp/part-001.NMZaTk/bin/maas-signal", line 227, in main
    power_pass=args.power_parms.split(",")[1],
IndexError: list index out of range
Success
```

Figure 5.1: MAAS IPMI error

The error shows that MAAS is having some problems while dealing with virtual machines. MAAS tries to adjust some power setting against the nodes. This was further investigated and problem found with ipmi tools. That leads to the same wake on lan challenge. The support of hardware is required for setting ipmi tools. Its seems that MAAS is setting up the machine by considering it as real machine and ends up in settings problems. There are some options in MAAS dashboard to start the node yourself. This issue could be related to that option.

Another to this explanation was found on this.

The failure that you see is because this is a virtual machine. The ipmisi related stuff will only work on real hardware, so the error doesn't really affect your case scenario and you don't really need a fix for it [49]

5.2.4 Issue: Issue 409 CONFLICT

After configuring the juju environment and running bootstrap command the error 409 occurred. It possibly means that juju had some problems while contacting to MAAS to get the prepared nodes. Unfortunately the official documentation[58] is not well explained regarding preparing the nodes at MAAS.

All the nodes prepared by MAAS, were assigned to the root user in MAAS dashboard. So when juju tries to contact MAAS for nodes, it does not get any node from MAAS. The issue is that all the nodes were assigned to root user in MAAS. So that is why juju was unable to find any free node. So the nodes should not be started second time without using Juju. The implemented solution is one way of doing it but it is possible to do it by not creating non admin user. This thesis tries to explore more about MAAS and its problems so non-admin user option was followed.

5.2.5 Issue: Juju status

The second solution entry in /etc/hosts fixes this issue. But this is not a professional approach to fix an issue. This will demand extra efforts in the future while deploying other nodes and the connection between those nodes. So is why this issue was ignored for the moment and was tried later in while dealing with a relation error. It is strange to see that a MAAS-DNS package is removed from the main repository and it is obsoleted. MAAS DNS was installed before during the installation procedures. Finding documentation[19] from MAAS team and noticing that they are using dnsmasq to fix these issues instead of MAAS-DNS. The reasons are explained briefly in result chapter.

5.2.6 Issue: Cobbler Errors

The found solution states to use time delay of 30 seconds but that does not work. Whereas making it to 60 seconds helps to fix this problem. This is was not a wise solution to go while deploying nodes at a larger scale. So this error should be fixed in some other way. Otherwise one could have continued with this way.

Considering this issue and some other issues like fault code 99 and ignored IPMI error, it is observed that MAAS does seem to have a good understanding with cobbler.

5.2.7 Issue Pending ID

Running Juju deploy command and wait finding which machine is assigned can be seen using an other way. MAAS dashboard changes does status change against any node if a node is assigned to some service. The status was changed for bootstrap node Steps were taken as directed at the official documentation [67] but it did not work. A delay of 15 minutes has been given by assuming that juju might be in the process of downloading the mysql and then assigning

instance ID. But still there is no change in instance ID. So it is obvious without knowing instance ID , one cannot simply guess that which node has been assigned to deploy mysql.

No good explanation is found over internet regarding this matter. Apparently, if juju can get one node for deploying the bootstrap node than it should be able to get more nodes for deploying requested services. Turning on some random machines make no sense and will lead to the issues. Running the juju status command in verbose mode gives some clues to solve this puzzle.

```
Juju -v status
```

From the output of above command, it is clear that juju only contacts bootstrap node to find the status. Moreover as said earlier, that the bootstrap node deploys the services on the other nodes. Which leads to a conclusion that bootstrap node is unable to find nodes. Whereas, it is possible that bootstrap node is using the same environment file configured before at MAAS server or it might be the case that the bootstrap node copy the contents of the environment file locally. The second scenario is copying the contents locally, shows that the bootstrap node is trying to find the nodes locally.

The official documentation of juju[61] asks to use localhost and that does not work. This should be fixed or some meaning full information should be added. But this might be the case that this documentation[61] which is still official documentation from Ubuntu has some issues so is why there is a broken link [60] towards juju documentation from main official documentation[61].

It was really hard to configure juju environment file for using MAAS as the official documentation does provide only the information on how to configure Juju for EC2.

Another way is to have a look at MAAS dashboard to notice any status change against any node as the status was changed for bootstrap node in previous section.

5.2.8 Issue fault code 99

Adding the next node without finishing the previous addition of node, lead to this issue. It can be very easily improved by adding validation in the web panel. Inserting some loading bar to show the process of the process of adding nodes can also be a good solution. Or this should be checked in a read system as well. The specification of MAAS system was changed and this issue was checked but this issue still exists. So the solution is to be patent while adding the nodes into MAAS dashboard. a gap of 2 second is enough.

5.2.9 Issue: block mapping

Nova volume needs a drive to put the images of instances. Openstack.cfg is the file used, in which pre-configuration were told. The file shown at official

documentation[67] asks uses ”xvdb” value for block mapping. It should be more meaningful to understand and to resolve the issue.

The errors produced by juju should also be more meaning as this was very time taking issue to get fixed. The number of spaces must be equal to four before starting the line block-device.

5.2.10 Challenge : booting nodes

After MAAS finishes installing the operating system the nodes stops at grub loader. Each node stops at the grub loader and asks for a human interaction to select an operating system to boot. Default option was selected. There is a possibility that the operating system was not installed properly on the nodes. This could lead to future problems and beside that, this challenge should be fixed to achieve as much automation as possible rather connecting every node and selecting from boot menu. This issue should be compared by implementing in a real environment that where is the problem. It could be a problem at the hypervisor.

To fix this issue the script start.pl [appendix D] was modified by making some time delays. Though the script does both automation of starting the nodes and fixing this issue but this issue should be further improved. Each node takes 30 mints to get prepared and then the next node starts for preparing. This is time consuming process to deploy a cloud. So the solution should be further improved.

5.2.11 Issue: Authentication

The step by step process discusses the error in detail in result chapter, as it is they are part of the results. So the error is not discussed here. But some other finding has been discussed here

After running the add relation command all the relations were created between services without any error. Juju status showed no error. But then luckily upon running juju status commands to check the status after 2 mints, it stated a relation error. This means juju was taking some time to create relations between the machines during this time. But then juju should state some meaningful information at upon status check. This should be improved by inserting information like pending (Juju uses this term when the nodes are in the process of operating system installation by MAAS). And once operating system is installed the status changes from pending to installed and then to ready state. Similar status reports should be applied for relations.

5.2.12 Issue: MAAS pxe boot issue

Upon investigation though the problem was found to be with tftp package missing but the question is why this problem occurred when MAAS was installed so many times before.

Findings

Changes have been made in MAAS but the official documentation[67] does not state these changes and does not explicitly guides to install tftp.

Upon reinstalling MAAS for solving a previous encountered issue, a new issue was discovered with MAAS. MAAS server did not respond to Pxe boot clients. To confirm that there is no problem while installing MAAS package or any other issue, the operating system was downloaded from ubuntu[56]. Because while installation of this operating system MAAS option can be selected. So it installs the MAAS automatically. After installation same problem was encountered.

5.2.13 Issue: Internal Server Error

After logging into to OpenStack dashboard this issue occurred. Juju status reported relation error and that is between nova-compute and nova cloud controller. Though by not using openstack.cfg solved the dashboard logging issue but the relation error still exists. Similarly when openstack.cfg was used for block device some issues were encountered. This means that nova cloud controller is not told to use Flat-DHCP for networking. This could be the possible reason of causing this issue.

The possible solution suggested above or in results chapter to fix this issue (placing openstack.cfg file at right path) did not work to solve the relation issue but it did help to fix other issue: using predefined password for OpenStack dashboard.

5.2.14 Issue: relation error

Juju has a debug command to fix problems but that did not fix this solution but from the log it was discovered that that nova-compute is having some DNS problems. The approach was to use a previously explored solution to another problem in this thesis. 4.1.4.2 Possible solution, Making an entry in /etc/hosts file by writing the first prepared node hostname with its IP

Possible solution was implemented with automaton. Step by steps implementation in section: Approach1 ”Appendix F” Following figure shows the

architecture

The relation error occurred again

Though this approach to fix this solution can be called wrong way or improper approach of solving this problem but it contributed to find a possible. After doing an in-depth study of nova compute it was discovered that the problem is possibly be with nova compute charm. It talks to DNS to resolve

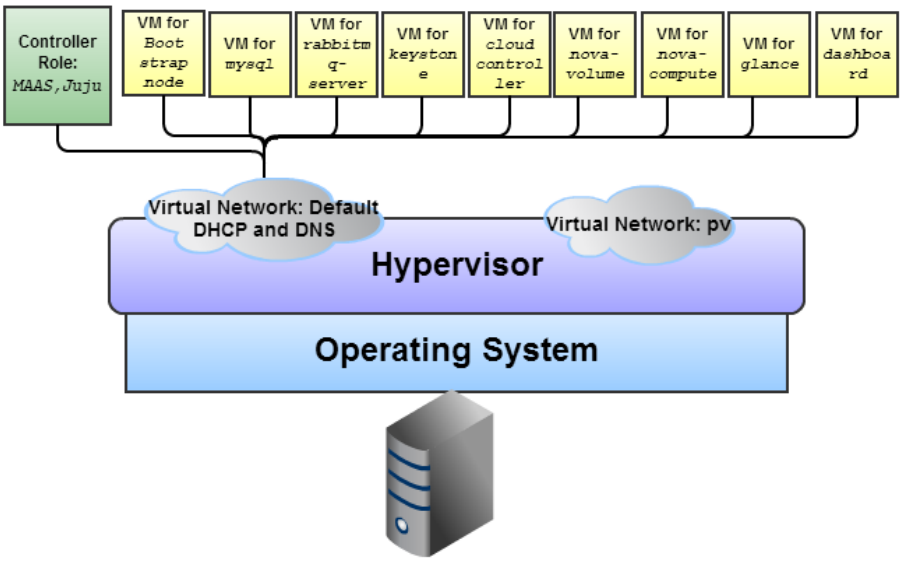

Figure 5.2: DHCP and DNS at KVM libvirt

the names but it does not support local dns resolution. nova-compute charm is written in a way, that it contacts Domain service directly. But it ignores local domain name translation. Whereas other services implemented has no such problem.

5.2.15 Issue: Juju charm

All of the add relation commands were executed. Juju status showed relation errors. Each node name includes the nodes MAC address without colon signs. e.g node-aabbccddeef0 DNS resolves these names to Ips successfully. Juju status shows an extra filed public address against each node. Public address: node-aabbccddeef0.localdomain

This documentation by MAAS team states

Findings: problem

Add the node using the MAC address, but for now specify the hostname to be 'node-¡mac address without colons¿' (eg node-aabbccddeef0). MAAS by default creates hostnames in this manner, but adds '.local'. We want to strip .local off of the hostname since dnsmasq does not seem to be working with avahi properly (FIXME) [19]

Above means it seems that MAAS team themselves are having this issue. So another approach was followed and that was to create separate virtual machine with domain controller and dhcp services. All this process was fully automated by creating the script resolver.pl [Appendix I].

Possible solution was implemented with automaton. Step by steps implementation in section: Approach2Appendix F. It was implemented such that

each virtual machine becomes a part of the domain malik.net. Hence it was expected that juju should generate public address of the machines somewhat like node-aabbccddeef0.malik.net but not node-aabbccddeef0.localdomain Following

figure shows the architecture

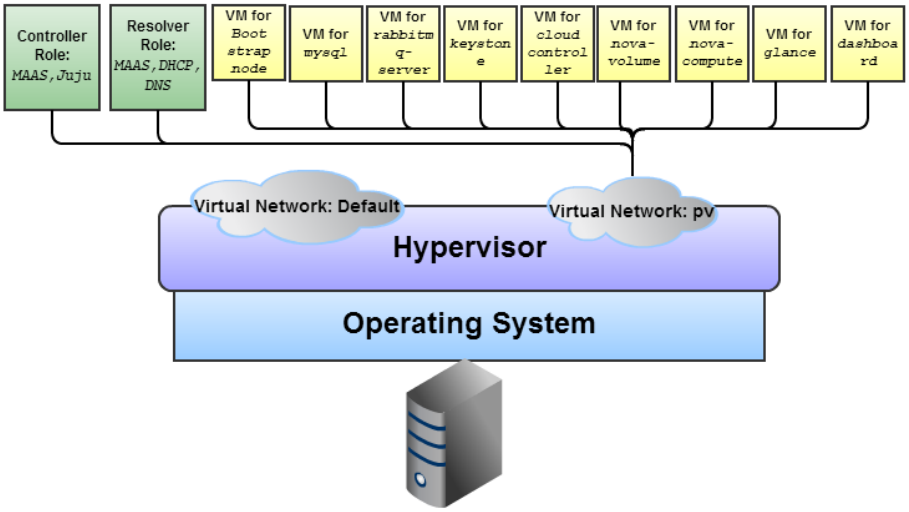

Figure 5.3: DHCP and DNS at seprate server

While doing this experiment a challenge was encountered:

5.2.15.1 Challenge: DHCP not found

Finding Nodes were unable to find DHCP server. The nodes first booted and then got the IP from MAAS server and got prepared by MAAS. But the next time when operating system installation started not were unable to find DHCP service in the network. Approach2Appendix F explains it more in detail

This was a strange problem to deal with because DHCP already did assigned the IP once when node booted for the first time. Practical solution suggests to unplug the network cable which is not possible. So thinking from real life scenarios the problem could be in network cable or network card or in DHCP service. Network cable was not possible to change as its virtual system and DHCP setting were fine so problem found was at node. The network card model did not work. Changing the network card model to rtl8139 solved this issue. The network card model selected before was Hypervisor default. This challenge was tested by destroying the whole environment and rebuilding it and making sure there is no other factor that is causing this problem. This can be explored and tested by the provided fully automated scripts for deploying a running dhcp and DNS serice.

5.2.16 Issue: Fixing Juju charm

All the systems were deployed as they were deployed before, but juju is attaching localdomain with the public addresses of the node names. But it should attach the created domain name malik.net. It is possibly leading to the same relation error, relation error between nova-cloud-controller and nova-compute. MAAS has been told in dashboard to use defined domain malik.net but juju still uses localdomain. Configuring Juju such that it knows the domain to be used is malik.net might fix this issue. Question[34] at ask Ubuntu has been generated and Jorge Castro,Deals with Ubuntu Cloud Infrastructure issues, has raised the bounty of the question to worth +250 reputations. Moreover six people did vote for this question, which apparently means that people are facing same issue.

Finding After investigating issue explained above, it is discovered that the relation error between nova compute and nova cloud controller is possibly because of DNS service. So to have more control over system DNS and DHCP service has been deployed on MAAS server using a script resolver.pl , rather keep trying the online documentations and fixing the solutions.

This script creates the domain by the name of localdomain and setups of DHCP service on MAAS server. After setting up domain and DCHP service on MAAS server MAAS and juju has been deployed and considering all the challenges faced earlier.

Though this approach removes the error and produces a running cloud. But the cloud created runs under the domain called localdomain. The question is not to produce a running cloud but to find the challenges and issues and fixing them. This was one way of fixing this issue. But a batter approach should be applied by finding how to use juju according for a different domain. The question at [34] has been update and it got +500 increase in reputation by Jorge Castro. But no answered found so far. The question [34] should help community.

5.3 Automation after applying solutions

Some scripts were discussed and analyzed in the previous section, here are the remaining scripts discussed and analyzed in detail.

After preparing MAAS the next step was to prepare nodes. This was done by writing a script nodecreater.pl[Appendix H] . It uses simple virsh commands to create and boot the machines from network. This script also helps to automate and fix some challenges. It also marks an entry in a text file after setting up a virtual machine. It does all the required automation but can be improved by using some variables.

Once the machines are in ready state in MAAS-dashboard Juju bootstrap command is executed manually and then the virtual machine, assigned to

bootstrap node, is started manually. This process is not automated because there is no assurance that MAAS will assign which machine to Juju bootstrap service. Usually it is the first machine that was prepared by MAAS. This can be automated by digging into MAAS and Jujus configuration files to see which machine has been assigned to bootstrap node and then turning that on inside the script.

The Services were deployed by running a script juju.pl [Apendix]. The script is executed at MAAS server. It first deploys executes all the commands for deployment and also the relation commands between the services.

The script start.pl has to be started to start all the remaining machines one by one, with some time delay in starting each of them. This script also helps to encounter the challenge 4.2.3. This script achieves the required automation for this project but it is not efficient enough for starting particular machines, because it starts all the machines. This can be done by improving the script juju.pl. So juju.pl may send the virtual machine names to be started, while juju.pl triggering this script. And on the other hand this script start.pl should read the arguments and should start the machines accordingly.

After starting each machine script start.pl marks an entry into a text file for system administrator easiness. After starting all the machines with the time delays it triggers the script start.pl at MAAS server. The IP of MAAS server has been used to avoid any DNS surprise.
. Another script ddbuilder.sh has been prepared to install and configure

DHCP and DNS

The machines resolver is prepared by using the same script creater.sh by doing little modifications and those modifications are changing the IP and name of the machine and copying script resolver.pl[app I] into the newly prepared machine resolver. The script resolver has should be configured according to system Ip and names. The script resolver.pl installs and configures the domain and DHCP services on the machine resolver and restarts the machine. It also configures DHCP so that the nodes are redirected to MAAS server for pxe boot.

Chapter 6

Conclusions

A step by step investigation has been made to find the possible challenges and issues while deploying Ubuntu cloud infrastructure in a virtual environment.

After finding each challenge and issues they were investigated to find a possible reason(s) of their occurrence.

Possible solutions were found by investigating the found reasons to the challenges and issues.

The found possible solutions were implemented to fix the challenges and issues. And where it was required more than one approach was followed to fix the issue or challenge.

Efforts have also been made to make automation scripts to build a fully automated virtual environment for deploying cloud. Most of the found solutions were implemented by doing automation. The produced scripts make it easier to avoid the encountered challenges and issues while deploying Ubuntu cloud infrastructure in a virtual environment.

In the light of findings of this thesis it is discovered that Ubuntu cloud infrastructure is not a simple, easy and fast solution to deploy in a virtual environment. But a contribution has been done to make it little faster and easier to deploy in a virtual environment by doing automation.

The idea, how MAAS and Juju should deploy a cloud, is found to be good. But various challenges and issues were faced while using MAAS and Juju in a virtual environment. The official documentations quality is not found to be good for deploying Ubuntu cloud infrastructure in a virtual environment. Whereas this thesis provides a step by step guide for the deployment of such a cloud by avoiding the possible challenges and issues.

6.1 Recommendations

If I am supposed to do the same task than I will try to fallow a different approach. I will use configuration management that may control the configuration. I will prefer to use a kickctart server on MAAS server. For deploying service on systems I will try juju-core and will retire the nodes from juju-core after they are prepared. I will try juju core by hoping that it will have less issues than Juju has. For creating relations between services or further managing these systems, I will use configuration management system.

Chapter 7

Future work

`Here is a list stating how this research can be continued`

1. juju-core and openstack grizelly
2. Implementing in real environment and comparing the results
3. Improving automation as directed in analysis and discussion chapter.
4. Replacing Juju with juju-core following the same scenario.
5. Implementing this thesis by using Ubuntu 12.10 or Ubuntu 12.13 operating system.
6. Deploying instances and finding the challenges and issues
7. Creating more virtual machines for adding compute nodes and finding challenges and issues
8. switching virtual network from natting mode to bridge mode and deploying some services on real machines.
9. Deploying different charms in a virtual environment.

Appendix A

builder.pl

```
#!/usr/bin/perl
 system("echo Installation started > installation_progress");
 system("apt-get update -y");
 system("sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder -y");
 system("rmmod kvm");
 system("modprobe -a kvm ");
 system("sudo apt-get install virt-manager -y ");
 system("sudo apt-get install qemu libcap2-bin bridge-utils -y ");
 system("apt-get install ubuntu-virt-server python-vm-builder -y");
 system("apt-get install kvm-ipxe virt-viewer -y");
 system("echo Configuring Bridge >> installation_progress");
  system("echo auto lo > /etc/network/interfaces");
  system("echo iface lo inet loopback >> /etc/network/interfaces");
  system("echo auto p2p1>> /etc/network/interfaces");
  system("echo iface p2p1 inet manual >> /etc/network/interfaces");
  system("echo auto bro >> /etc/network/interfaces");
  system("echo iface bro inet static >> /etc/network/interfaces");
  system("echo address 10.0.0.4 >> /etc/network/interfaces");
  system("echo network 10.0.0.0 >> /etc/network/interfaces");
  system("echo netmask 255.255.255.0 >> /etc/network/interfaces");
  system("echo gateway 10.0.0.1 >> /etc/network/interfaces");
  system("echo bridge_ports p2p1 >> /etc/network/interfaces");
  system("echo bridge_stp off >> /etc/network/interfaces");
  system("echo bridge_fd 0 >> /etc/network/interfaces");
  system("echo bridge_maxwait 0 >> /etc/network/interfaces");
  system("echo dns-nameservers 10.0.0.1 >> /etc/network/interfaces");
  system("echo Restarting Network >> installation_progress");
  system("/etc/init.d/networking restart");
  system("echo Restarting system >> installation_progress");
system("./network.sh");
 system("sudo reboot");
```

Appendix B

network.sh

```
echo "<network>" > /var/lib/libvirt/network/pv.xml
echo "<name>pv</name>" >> /var/lib/libvirt/network/pv.xml
echo "<uuid>ea2fe5d8-9b01-badd-d2d2-7f8aa91f1db6</uuid>" >> /var/lib/libvirt/network/pv.xml
echo "<forward mode='nat'/>" >> /var/lib/libvirt/network/pv.xml
echo "<bridge name='virbr1' stp='on' delay='0' />" >> /var/lib/libvirt/network/pv.xml
echo "<mac address='52:54:00:8E:5A:4E'/>" >> /var/lib/libvirt/network/pv.xml
echo "<ip address='192.168.100.1' netmask='255.255.255.0'>" >> /var/lib/libvirt/network/pv.xml
echo "</ip>" >> /var/lib/libvirt/network/pv.xml
echo "</network>" >> /var/lib/libvirt/network/pv.xml
virsh net-define /var/lib/libvirt/network/pv.xml
virsh net-start pv
virsh net-autostart pv
```

Appendix C

creater.sh

```
mkdir -p /var/lib/libvirt/images/vm1/mytemplates/libvirt
cp /etc/vmbuilder/libvirt/* /var/lib/libvirt/images/vm1/mytemplates/libvirt/
cd /var/lib/libvirt/images/vm1/
echo root 9000 > vmbuilder.partition
echo swap 4000 >> vmbuilder.partition
echo ---- >> vmbuilder.partition
echo /var 20000 >> vmbuilder.partition
echo > boot.sh
echo apt-get update >> boot.sh
echo apt-get install -qqy --force-yes openssh-server >> boot.sh
vmbuilder kvm ubuntu --suite=precise --flavour=virtual --arch=amd64
mirror=http://de.archive.ubuntu.com/ubuntu -o
--libvirt=qemu:///system --ip=192.168.100.7 --gw=192.168.100.1
--part=vmbuilder.partition --templates=mytemplates
--user=adnan --name=adnan --pass=120476
--addpkg=vim-nox --addpkg=unattended-upgrades
 --addpkg=acpid --firstboot=/var/lib/libvirt/images/vm1/boot.sh
--mem=1000 --hostname=controller --network=pv --cpus=1
virsh start controller
iptables -I FORWARD -m state -d 192.168.100.0/24 --state NEW,RELATED,ESTABLISHED -j ACCEPT
iptables -t nat -I PREROUTING -p tcp -i br0 --dport 80 -j DNAT --to-destination 192.168.100.7:80
sleep 30
apt-get install sshpass -y
sshpass -p '120476' scp -o StrictHostKeyChecking=no guest/* adnan@192.168.100.7:/home/adnan/
```

Appendix D

start.pl

```
#!/usr/bin/perl
system("virsh start n1");
system("echo n1 >> progress");
 sleep 1800;
system("virsh start n2");
system("echo n2 >> progress");
 sleep 1800;
system("virsh start n3");
system("echo n3 >> progress");
 sleep 1200;
system("virsh start n4");
system("echo n4 >> progress");
 sleep 1200;
system("virsh start n5");
system("echo n5 >> progress");
 sleep 1200;
system("virsh start n6");
system("echo n6 >> progress");
sleep 1200;
system("virsh start n7");
system("echo n7 >> progress");
sleep 1200;
system("virsh start n8");
system("echo n8 >> progress");
 sleep 1200;
# system("virsh start n9");
# system("echo n9 >> progress");
# sleep 300;
system("echo Done >> progress");
```

Appendix E

approach1: local DNS

E.0.1 Using Libvirts DHCP and DNS

E.0.1.1 Approach

Libvirt has its own DHCP and DNS services that can be used for at least resolving DNS of system names. And later /etc/hosts files at each virtual machine shall be modified by writing the public addresses generated by juju against the IP of each system.

Virtual network default has these services enabled by default. MAAS and Juju will deal with preparing the nodes and deploying the services on these nodes. First MAAS server shall be prepared on a virtual machine. DHCP at Libvirt will be configured such that all the pxe boot clients are directed to MAAS server. This case will also help to check whether it is possible to make Ubuntu Cloud infrastructure work with Libvirts provided services or not.

The process of creating and configuring virtual machines will be similar to as explained in section before, but with these differences.

- Virtual network default shall be used for all the guest machines
- DHCP and DNS shall not be installed on MAAS Server.
- Libvirt will assign the IPs to the nodes and will redirect the pxe boot nodes to the MAAS server

Following figure shows the architecture

Server Technical Environment

The following tables show the basic information of the systems that will be used. Virtual network interface for the virtual machines will be virbr0 and network default.

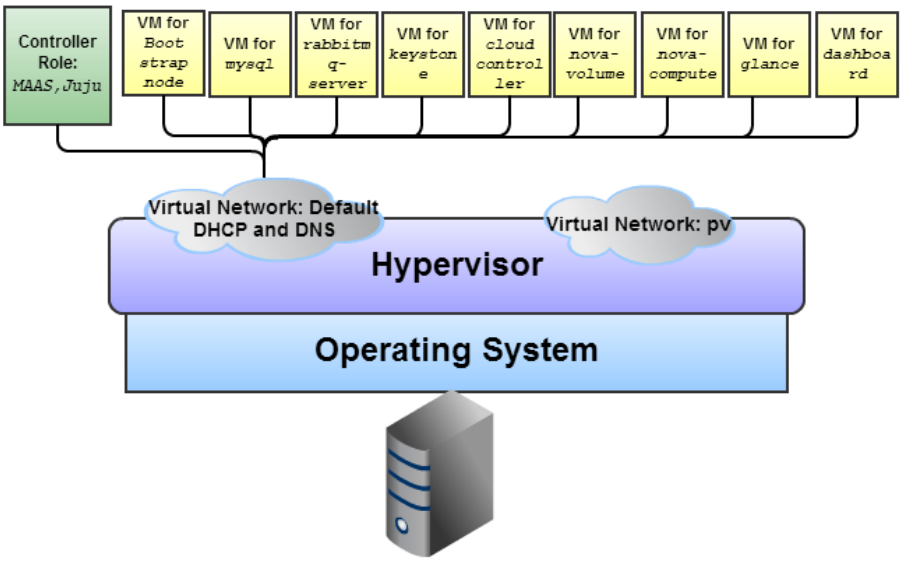

Figure E.1: DHCP and DNS at KVM libvirt

HOSTNAME	IP ADDRESS	SYSTEM TYPE	ROLE
Trident	192.168.122.1	Real	Host system
Controller	192.168.122.7	Virtual	MAAS and Juju

Table E.1: Systems in case:1

E.0.1.2 Implementation

A virtual machine has been prepared to deploy MAAS server as guided above. MAAS and Juju has been installed successfully by using the same procedure as it was done in previous section, but with an exception of not installing MAAS-DNS and MAAS-DHCP.

The solution to issue: juju-status has been implemented here by edition the /etc/hosts file of MAAS server. The string localdomain is attached by MAAS or juju. So the DNS is unable to translate this public address to IP. This could possibly be the problem. To fix this issue some entries are made in "/etc/hosts" file at each node. So that the nodes could translate the public address locally.

A sample entry is as below

Listing E.1: /etc/hosts

```
192.168.122.6   node-aabbccddeef0.localdomain
192.168.122.7   node-aabbccddeef1.localdomain
```

Only first two entries are shown above. Similarly against each nodes IP, its public address has been written. These entries are made into the nodes "/etc/hosts file". nmap command has been used to find the IPs assigned to each node.

E.0.1.3 Result

The relations between the nodes were create but landed to the same relation error problem between nova-cloud controllers an nova compute. So this solution does not work.

Appendix F

approach2: Separate domain controller

F.0.2 Case 2: DHCP and DNS on a separate virtual machine

F.0.2.1 Approach

First a virtual machine shall be created to deal with DNS and DHCP services. After that a second machine will be created, which will get IP from this first virtual machine and it also will get registered under the domain created in the first machine. This second machine shall be configured as MAAS server. Similarly further virtual machines shall be treated.

Here is how it should work.

- First the virtual machine will be prepared for DNS and DHCP server.
- Second the virtual machine will be prepared for MAAS server, but this machine will get IP from the first machine.
- DHCP and DNS services shall not be installed on MAAS Server.
- First machine with DNS and DHCP configured will assign the Ips to the nodes and will redirect the pxe boot nodes to the MAAS server

Server Technical Environment

The following tables show the basic information of the systems that will be used during case 2. Virtual network interface for the virtual machines will be virbr0 and network pv.

HOSTNAME	IP ADDRESS	SYSTEM TYPE	ROLE
Trident	192.168.100.1	Real	Host system
resolver	192.168.100.5	Virtual	DHCP and DNS
Controller	DHCP	Virtual	MAAS and Juju

Table F.1: Systems in case:2

F.0.2.2 Implementation

DNSMASQ is used to provide DHCP and DNS services. A script resolver.pl[app I] has been written which installs dnsmasq and then it configures the DNS and DHCP services. Once this machine is up and running, MAAS server is configured as explained in approach. System deployment started challenge encountered ? MAAS and Juju as has been installed successfully by using the same procedure as it was done in previous section, but with an exception of not installing MAAS-DNS and MAAS-DHCP. Moreover a little change has been done in MAAS dashboard. Under settings section, the domain name has been specified to malik.net.

All the nodes in MAAS dashboard are in ready state. Next step is to prepare bootstrap node. Upon turning on the node, it contacts DHCP server "resolver.malik.net' and get the IP and is redirected to controller.malik.net for operating system installation.

CHALLENGE

Operating system installation starts successfully on boot strap node but then it stops at getting the IP from DHCP server.

F.0.3 Fixing DHCP challenge and continuing

DHCP server has been checked thoroughly but no problem has been found at DHCP sever. Some online solution recommends unplugging network cable, which is simply not possible in a virtual network. Upon changing the network card model and restating the bootstrap nodes solves the challenge. It has been discovered that the default assigned network model does not work in getting the IP from separate hosted dnsmasq based DHCP. The network bridge option has been extended while creating the virtual machines for nodes. Model property of Network Bridge was not used before.

The command in the script now becomes.

```
virt-install --name n0 --ram 777 --arch x86_64 --vcpus 1
--os-type=linux --os-variant=virtio26
--file=/var/lib/libvirt/images/n0.img --file-size=15
--graphics vnc --accelerate --hvm --network
bridge:virbr1,model=rtl8139 --pxe --mac=aa:bb:cc:dd:ee:f0
```

After preparing the juju bootstrap node, all the juju deployment commands were executed. The script start.pl [appendix D] is executed to start all the nodes for operating system installation. Now the nodes are in the process of operating system installation. Now all of the nodes are getting registered under a domain malik.net. So now juju should create the public address something like node-aabbccddeef0. malik.net but not node-aabbccddeef0.localdomain.

ISSUE

All the systems were deployed as they were deployed before, but juju is attaching localdomain with the public addresses of the node names. It is possibly leading to the same relation error, relation error between nova-cloud-controller and nova-compute. MAAS has been told in dashboard to use defined domain malik.net but juju still uses localdomain. Configuring Juju such that it knows the domain to be used is malik.net might fix this issue.

Appendix G

maas1.sh

```
#!/bin/sh
sudo apt-get install maas debmirror -y
sudo apt-get install maas-enlist tftpd-hpa -y
sudo maas-import-isos
sudo apt-get update && sudo apt-get install juju charm-tools -y
juju bootstrap
cp  environments.yaml  .juju/environments.yaml
sudo reboot
```

Appendix H

nodecreater.pl

```
#!/usr/bin/perl
system("virt-install --name n0 --ram 700 --arch x86_64 --vcpus 1 --os-type=linux --os-variant=virtio26 --file=/var/lib/libvirt/images/n0.
system("echo n0 > progress");
 sleep 20;
system("virt-install --name n1 --ram 700 --arch x86_64 --vcpus 1 --os-type=linux --os-variant=virtio26 --file=/var/lib/libvirt/images/n1.
system("echo n1 started >> progress");
 sleep 20;
system("virt-install --name n2 --ram 777 --arch x86_64 --vcpus 1 --os-type=linux --os-variant=virtio26 --file=/var/lib/libvirt/images/n2.
system("echo n1 started >> progress");
 sleep 20;
 system("virt-install --name n3 --ram 774 --arch x86_64 --vcpus 1 --os-type=linux --os-variant=virtio26 --file=/var/lib/libvirt/images/n3
 system("echo n3 started >> progress");
 sleep 20;
 system("virt-install --name n4 --ram 774 --arch x86_64 --vcpus 1 --os-type=linux --os-variant=virtio26 --file=/var/lib/libvirt/images/n4
 system("echo n4 started >> progress");
 sleep 20;
system("virt-install --name n5 --ram 774 --arch x86_64 --vcpus 1 --os-type=linux --os-variant=virtio26 --file=/var/lib/libvirt/images/n5.
 system("echo n5 started >> progress");
 sleep 20;
 system("virt-install --name n6 --ram 774 --arch x86_64 --vcpus 1 --os-type=linux --os-variant=virtio26 --file=/var/lib/libvirt/images/n6
 system("echo n6 started >> progress");
 sleep 20;
 system("virt-install --name n7 --ram 724 --arch x86_64 --vcpus 1 --os-type=linux --os-variant=virtio26 --file=/var/lib/libvirt/images/n7
 system("echo n7 started >> progress");
 sleep 20;
 system("virt-install --name n8 --ram 724 --arch x86_64 --vcpus 1 --os-type=linux --os-variant=virtio26 --file=/var/lib/libvirt/images/n8
 system("echo n8 started >> progress");
 sleep 20;
 system("virt-install --name n9 --ram 724 --arch x86_64 --vcpus 1 --os-type=linux --os-variant=virtio26 --file=/var/lib/libvirt/images/n9
 system("echo n9 started >> progress");
 sleep 20;
 system("echo All of the machines should be in Ready state, please check in MAAS dashboard >> progress");
```

Appendix I

resolver.pl

```
#!/bin/sh
sudo apt-get install nano dnsmasq -y
echo no-hosts > /etc/dnsmasq.conf
echo addn-hosts=/etc/hosts.dnsmasq >> /etc/dnsmasq.conf
echo server=/localnet/192.168.100.7 >> /etc/dnsmasq.conf
echo server=/#/8.8.8.8 >> /etc/dnsmasq.conf
echo server=/#/8.8.4.4 >> /etc/dnsmasq.conf
echo domain=localdomain >> /etc/dnsmasq.conf
echo dhcp-range=192.168.100.8,192.168.100.150,48h >> /etc/dnsmasq.conf
echo dhcp-option=option:router,192.168.100.1 >> /etc/dnsmasq.conf
echo dhcp-authoritative >> /etc/dnsmasq.conf
echo ptr-record=7.100.168.192.in-addr.arpa.,"controller.localdomain" >> /etc/dnsmasq.conf
echo address=/controller.localdomain/192.168.100.7 >> /etc/dnsmasq.conf
echo dhcp-boot=pxelinux.0,server,192.168.100.7 >> /etc/dnsmasq.conf
sudo touch /etc/hosts.dnsmasq
sudo service dnsmasq restart
sudo reboot
```

Appendix J

juj.pl

```
#!/bin/sh
juju deploy mysql
juju deploy rabbitmq-server
juju deploy --config=openstack.cfg keystone
juju deploy --config=openstack.cfg nova-cloud-controller
juju deploy --config=openstack.cfg nova-volume
juju deploy nova-compute
juju deploy glance
juju deploy openstack-dashboard

juju add-relation keystone mysql
sleep 120;
juju add-relation nova-cloud-controller mysql
sleep 120;
juju add-relation nova-cloud-controller rabbitmq-server
sleep 120;
juju add-relation nova-cloud-controller glance
sleep 120;
juju add-relation nova-cloud-controller keystone
sleep 120;
juju add-relation nova-volume nova-cloud-controller
sleep 120;
juju add-relation nova-volume mysql
sleep 120;
juju add-relation nova-volume rabbitmq-server
sleep 120;
juju add-relation nova-compute mysql
sleep 120;
juju add-relation nova-compute rabbitmq-server
sleep 120;
juju add-relation nova-compute glance
sleep 120;
juju add-relation nova-compute nova-cloud-controller
sleep 120;
juju add-relation glance mysql
sleep 120;
juju add-relation glance keystone
sleep 120;
juju add-relation openstack-dashboard keystone
juju expose openstack-dashboard
juju expose nova-cloud-controller
```

Bibliography

[1] Ajit. Metal as a service: Maas for cloud deployment on ubuntu precise panligon. Available at http://fafadiatech.blogspot.no/2012/04/metal-as-service-maas-for-cloud.html, 2013 March.

[2] Wikimedia Commons. Cloud computing. Available at http://en.wikipedia.org/wiki/File:Cloud_computing_layers.png [Accessed 4.03.2013], 2013.

[3] S Cosgrove. Bring together a low-cost networking learning environment. *ACM SIGITE*, West Point, NY:101–107, 2011.

[4] China Daily. Five cities chosen for cloud computing pilot plan. Available at www.chinadaily.com.cn/cndy/2011-11/18/content_14115178.htm [Accessed 01.03.2013], 2011.

[5] Peoples Daily. Shanghai releases 3-year cloud computing plan. Available at http://english.peopledaily.com.cn/90001/90778/90860/7108660.html [Accessed 01.03.2013], 2011.

[6] Daniel and Julian Edwards. Unable to create node: timed out. Available at https://launchpad.net/~julian-edwards, April 2013.

[7] Charmworld Developers. Charm browser. Available at http://jujucharms.com/charms [Accessed 02.06. 2013], 2013.

[8] Diablo. Components of openstack compute. Available at http://docs.openstack.org/diablo/openstack-compute/starter/content/Components_of_C [Accessed 10.03.2013], 2013.

[9] Digit. History of cloud computing: Timeline. Available at ;http://sourcedigit.com/497-timeline-history-of-cloud-computing [Accessed 28.02.2013], 2012.

[10] Julian Edwards and Jorge Castro. unable to create node: timed out error when creating node in maas. Available at http://askubuntu.com/questions/128322/unable-to-create-node-timed-out-error-when- 2013 April.

[11] EMC. White paper: Virtualizing business-critical applications. EMC, 2010.

[12] eNovance. The plateform for scale out. Available at http://www.slideshare.net/enovance/open-stack-in-action2-canonical-openstack-clou [Accessed 24. 01. 2013], 2002.

[13] falko. Virtualization with kvm on ubuntu 12.04 lts. Available at http://www.howtoforge.com/virtualization-with-kvm-on-ubuntu-12.04-lts [Accessed 05.03. 2013], 2013.

[14] Yahoo Finance. China cloud programme will more than double datacenter capacity with investment of usd 370 billion. Available at ;http://uk.finance.yahoo.com/news/china-cloud-programme-more-double-110200676.htm [Accessed 01.03.2013], 2011.

[15] Grizzly. Components of openstack. Available at http://docs.openstack.org/trunk/openstack-compute/admin/content/components-of-oper [Accessed 8.03.2013], 2013.

[16] InternetSlang. The slang word / acronym / abbreviation juju. Available at http://www.internetslang.com/JUJU-meaning-definition.asp [Accessed 19. 05. 2013], 2013.

[17] W. Pitt Turner IV and Kenneth G. Brill. Cost model: Dollars per kw plus dollars per square foot of computer floor. Uptime Institute, 2008.

[18] Kevin Jackson. Installing openstack services using juju. Available at http://my.safaribooksonline.com/book/-/9781849517324/11dot-in-the-datacenter/id28 April 2013.

[19] jdstrand. Securityteam/testingmaas. Available at https://wiki.edubuntu.org/SecurityTeam/TestingMAAS [Accessed 02.05. 2013], 2013.

[20] K. Jeffery and B. Neidecker-Lutz. The future of cloud computing. Available at http://cordis.europa.eu/fp7/ict/ssai/docs/cloud-report-final.pdf [Accessed 01.03.2013], 2011. Opportunities for European Cloud Computing Beyond 2010.

[21] Bojanova Irena Zhang Jia Voas Jeffrey. Cloud computing. *IT Professional*, 15 , Issue: 2:12 – 14, 2013.

[22] Fern Halper Dan Kirsch Judith Hurwitz, Marcia Kaufman. What is cloud computing? *Paperback*, ISBN: 978-1-118-12719-3:360, 2012.

[23] kirkland. Kvm/virsh. Available at https://help.ubuntu.com/community/KVM/Virsh [Accessed 20.02.2013], 2013.

[24] Martijn Koster. Metal as a service. Available at http://www.greenhills.co.uk/2012/04/25/metal-as-a-service.html [Accessed 10.03. 2013], 2013.

[25] Lucian Popa Sylvia Ratnasamy Gianluca Iannaccone Arvind Krishnamurthy and Ion Stoica. A cost comparison of data center network architectures. University of California, Berkeley, 2010.

[26] V. Kundra. State of public sector cloud computing. Available at www.cio.gov/pages.cfm/page/State-of-Public-Sector-Cloud-Computing [Accessed 01.03.2013], 2011. US Chief Information Officers Council, 2010.

[27] Li P Toderick L. Cloud in cloud- approaches and implementations. *ACM SIGITE*, Midland, Michigan:105–110, 2010.

[28] P Li. Selecting and using virtualization solutions- our experiences with vmware and virtual box. Consortium for Computing Sciences in Colleges, 2010. 11-.

[29] P Li. Introducing virtualization management concepts using open source cloud infrastructure managers. *SIGITE*, 0:309–310, 2011.

[30] libvirt. Libvirt networking. Available at http://wiki.libvirt.org/page/Networking [Accessed 22.02.2013], 2013.

[31] libvirt. Virsh command reference. Available at http://libvirt.org/virshcmdref.html [Accessed 17.02.2013], 2013.

[32] libvirt. The virtualization api. Available at http://libvirt.org/ [Accessed 22.02.2013], 2013.

[33] Holden EKang J Bills D IIyassov M. Database in the cloud: a work in progress. *SIGITE 09*, Fairfax, VA:138–143, 2009.

[34] Adnan Malik. relation error between nova-compute and nova cloud controller. Available at http://askubuntu.com/questions/291736/relation-error-between-nova-compute-and-nov May 2013.

[35] SCOTT MERRILL. Canonical metal-as-a-service. Available at http://techcrunch.com/2012/04/04/canonical-metal-as-a-service-not-quite-as-cool-a [Accessed 24. 03. 2013], 2013.

[36] A. Mohamed. A history of cloud computing. Available at www.computerweekly.com/feature/A-history-of-cloud-computing [Accessed 28.02.2013], 2009.

[37] R. Moreno-Vozmediano. Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures. *IEEE Computer Society*, Volume:45 , Issue: 12:65–72, 2012.

[38] Shenzhen Government Online. Tencent starts cloud computing project. Available at http://english.sz.gov.cn/ln/201112/t20111219_1788830.htm [Accessed 01.03.2013], 2011.

[39] openstack. Hypervisors. Available at http://www.openstack.org/blog/tag/hypervisors/ [Accessed 16.02.2013], 2013.

[40] openstack. Openstack: The open source cloud operating system. Available at http://www.openstack.org/software/ [Accessed 05.02. 2013], 2013.

[41] Siani Pearson. Taking account of privacy when designing cloud computing services. HP Laboratories, 2009.

[42] Ken Pepple. Ubuntu cloud infrastructure. Available at http://www.ubuntu.com/cloud/public-cloud/infrastructure [Accessed 13.03.2013], 2013.

[43] L Powell V Johnson R Turchek J Davis C & Parker. Vlabnet: The integrated design of hands-on learning in information security and networking. In *Information Security Curriculum Development Conference07*, pages 1–7, 2007.

[44] Bangkok North. Bangkok Vannarat S PrueksaaroonS North. An implementation of virtualization cluster: Extending beowulf cluster using virtualization cluster management and image storage. *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. 6th International Conference on*, 02:700 – 703, 2009.

[45] Rackspace. Open source software for building private and public clouds. Available at http://www.openstack.org/ [Accessed 07.03.2013], 2013.

[46] Forrester Research. Virtual security in the data center. Forrester Research, 2012.

[47] Thomas Ritzau Robert Warnke. qemu-kvm & libvirt. ISBN 978-3-8370-0876-0.

[48] M. Rosenblum and T. Garfinkel. Virtual machine monitors: Current technology and future trends, apr 2005. 39-47.

[49] Ubuntu Developer Sandor. Maas - fatal error inserting. Available at http://askubuntu.com/questions/298167/maas-fatal-error-inserting-ipmi-si, may 2013.

[50] StephaneKlein. vmbuilder. Available at http://wiki.debian.org/VMBuilder [Accessed 24.02.2013], 2013.

[51] Stewart K Humphries Humphries J Andel T. Developing a virtualization platform for courses in networking, systems administration and cyber security education, 2008.

[52] Falko Timme. Virtualization with kvm on ubuntu 12.04 lts. Available at http://www.howtoforge.com/virtualization-with-kvm-on-ubuntu-12.04-lts [Accessed 24.02.2013], 2012.

[53] ubuntu. maas-cli commands. Available at http://maas.ubuntu.com/docs/maascli.html#cli-power [Accessed 01.03.2013], 2011.

[54] ubuntu. Charm: nova-volume. Available at http://jujucharms.com/charms/precise/nova-volume, march 2013.

[55] Ubuntu. Devops distilled. Available at https://juju.ubuntu.com/ [Accessed 4.04.2013], 2013.

[56] ubuntu. Download ubuntu server for cloud. Available at http://www.ubuntu.com/download/cloud, Feburary 2013.

[57] Ubuntu. Iaas we can! ubuntu cloud infrastructure. Available at http://www.ubuntu.com/content/iaas-we-can-ubuntu-cloud-infrastructure [Accessed 20.03.2013], 2013.

[58] ubuntu. Installing maas. Available from http://maas.ubuntu.com/docs/install.html, March 2013.

[59] Ubuntu. Jeos and vmbuilder. Available at https://help.ubuntu.com/12.04/serverguide/jeos-and-vmbuilder.html [Accessed 24.02.2013], 2013.

[60] Ubuntu. Juju. Available at https://wiki.ubuntu.com/ServerTeam/MAAS/Juju [Accessed 05.03.2013], 2013.

[61] Ubuntu. Juju quick start. Available at https://maas.ubuntu.com/docs/juju-quick-start.html, march 2013.

[62] Ubuntu. Kvm installation. Available at https://help.ubuntu.com/community/KVM/Installation [Accessed 02.03. 2013], 2013.

[63] Ubuntu. Metal as a service. Available at https://maas.ubuntu.com/ [Accessed 28.03.2013], 2013.

[64] Ubuntu. Openstack with ubuntu. Available at http://www.ubuntu.com/cloud/private-cloud/openstack [Accessed 22.03.2013], 2013.

[65] Ubuntu. Private cloud. Available at http://www.ubuntu.com/cloud/private-cloud [Accessed 19. 05. 2013], 2013.

[66] Ubuntu. Adding nodes to the system. Available at https://maas.ubuntu.com/docs/nodes.html, 2013 April.

[67] Ubuntu. Ubuntucloudinfrastructure. Available at https://help.ubuntu.com/community/UbuntuCloudInfrastructure, 2013 March.

[68] virsh. virsh management user interface. Available at http://manpages.ubuntu.com/manpages/karmic/en/man1/virsh.1.html [Accessed 20.02.2013], 2013.

[69] Z. Walton. Americans think cloud computing comes from actual clouds. Available at www.webpronews.com/americansthink-cloud-computing-comes-from-actual-clouds/ [Accessed 28.02.2013], 2012.

[70] G Cerier Perrault Wang X, Hembroff. Introducing cloud computing with a senior design project in undergraduate education of computer system and network administration. *ACM SIGITE*, West Point, USA:177–182, 2011.

[71] Zhen Xiao. Dynamic resource allocation using virtual machines for cloud computing environment. *Parallel and Distributed Systems, IEEE Transactions*, 24 , Issue 6:1107–1117, 2013.

[72] Stackpole B Koppe J Haskell T Guay L. Pan Y. Decentralized virtualization in systems administration education. *ACM SIGITE*, Cincinnati, Ohio:249–255, 2008.

[73] R. Buyya CS Yeo and S. Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. *Proc. IEEE Intl Conf. High Performance Computing and Communications*, HPCC 08:. 5–13, 2008.

[74] C Yuan, D.Q. & Lewandowski. Developing an ip telephony laboratory and curriculum with private cloud computing. *ACM SIGITE*, West Point,NY:171–177, 2011.